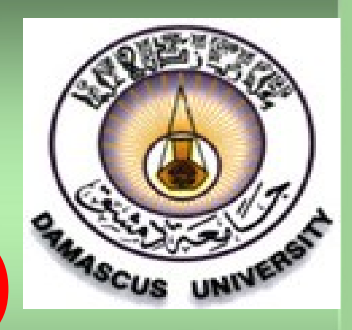


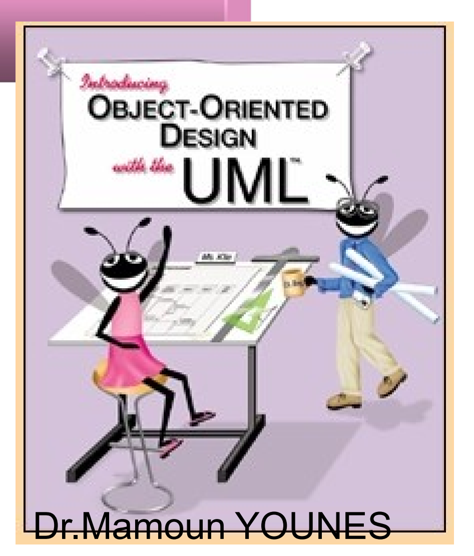
البرمجة(1) بلغة C#
Programming(1)



Dr. Mamoun Younes







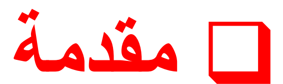


المكونات الأساسية للغة C# وأدواتها

Basic Elements Of C#

Chapter 1

محتويات الفصل الأول



❑ مقدمة

- اللغات البنيوية
- اللغات غرضية التوجه



❑ بنية البرنامج في لغة #C

- التعليقات
- فضاءات الأسماء
- التابع ( )Main
- توابع الإدخال والإخراج
- متواليات الهروب

- تنسيق الخرج
- تعليمة الإدخال
- إدخال المعطيات المختلفة
- أمثلة
- الأسئلة

❑ رموز لغة C#
- العوامل الحسابية
- العوامل العلائقية والمنطقية
- أولوية العمليات الحسابية

❑ أنواع المعطيات الأساسية في لغة C#
- المتحولات المحرفية char
- المتحولات الصحيحة
- المتحولات الحقيقية
- الأسئلة

- العوامل الحسابية
- العوامل العلائقية والمنطقية
- المعامل ثلاثي الحدود
- أولوية العمليات الحسابية
- أمثلة
- الأسئلة

مقدمة

- تعتبر لغة C# من أشهر اللغات التي تتمتع بطابع القوة والمرونة لإنتاج أسرع البرامج وأفضلها أداءاً.
- وعلى الرغم من وجود العديد من لغات البرمجة الأخرى إلا أنها تفتقر شمولية لغة C# وقوتها .
- فاللغة C# تتميز بقابليتها على معالجة التطبيقات الكبيرة والمعقدة، والقوة في صيانة البرامج المكتوبة بها مما يوفر وقتاً في تصميم البرامج وتطويرها .
- تعتبر لغة C# لغة برمجة غرضية التوجه طورتها شركة Microsoft ، وتستخدم هذه اللغة منصة عمل (.NET) وبالتالي تملك مجموعة واسعة من المكونات البرمجية ، مما يجعلها أداة برمجية خصبة جداً .
- يمكن باستخدام C# أن ننشئ وبسرعة تطبيقات Windows ، أو تطبيقات ويب أو تطبيقات طرفيات أو برامج شبكة أو تطبيقات قواعد البيانات أو مكتبات برمجية
- بما أن لغة C# لغة برمجة غرضية التوجه ، مما يعني أن برنامج C# يتكون من مجموعة من الأغراض Objects التي تتواصل مع بعضها بعضاً في زمن التشغيل.

- يتم توصيف هذه الأغراض بواسطة الصفوف Classes يتم تعريفها عند كتابة البرنامج .
- كما أن لغة C# تمتلك ميزات استثنائية لمعالجة الخطأ وإدارة مؤتمتة للذاكرة وتدعى هذه الميزة بجمع النفايات garbage collection وكل هذا لتسهيل تطوير التطبيق .
- تعتمد اللغة C# أسلوب البرمجة غرضية التوجه Object Oriented Programming، والذي يعرف اختصاراً بـ (OOP)، والذي تم تطويره بسبب قيود كانت أساليب البرمجة القديمة المتمثلة في اللغات البنيوية تفرضها على المبرمجين.
- ولكي نتعرف على طبيعة تلك القيود يجب أن نلقى الضوء على ما يحدث في اللغات البنيوية.

اللغات البنيوية

- لغات Pascal، C، Basic و Fortran وغيرها من لغات البرمجة التقليدية هي لغات بنيوية أو إجرائية (Procedural). أي أن كل عبارة في اللغة هي عبارة عن تعليمة للحاسوب أن ينفذ شيئاً ما : للحصول على دخل أو جمع أرقام الخ.. ... . لذا نجد أن البرنامج المكتوب بلغة بنيوية هو عبارة عن لائحة من التعليمات.
- لا تبدو هنالك مشكلة مع البرامج الإجرائية الصغيرة، فالمبرمج ينشئ لائحة التعليمات ويقوم الحاسوب بتنفيذها. ولكن مع كبر حجم البرامج لا تعود لائحة من التعليمات فعالة حيث يصعب فهم برنامج يتألف من مئات من العبارات .
- لذا تم اعتماد أسلوب التوابع (Functions) والإجراءات (Procedures) كوسيلة لجعل البرامج أسهل للقراءة والفهم، حيث يمتلك كل تابع في البرنامج واجهة محددة، وينفذ هدفاً محدداً.

- ولكن المشكلة ما تزال قائمة : مجموعة من التعليمات تنفذ مهاماً محددة. و مع تزايد حجم البرامج وتعقيدها، يظهر ضعف الأسلوب الإجرائي، حيث تصبح البرامج الضخمة معقدة إلى حد كبير .
- غالباً ما يكون تصميم البرامج الإجرائية صعباً، لأن مكوناتها الرئيسية (التوابع) عبارة عن بنية معطيات لا تقلد العالم الحقيقي جيداً. و يصعب في اللغات الإجرائية إنشاء أي نوع معطيات جديد بخلاف الأنواع المعرفة أصلاً في تلك اللغات ، لكل هذه الأسباب تم تطوير لغات غرضية التوجه .

لغات البرمجة غرضية التوجه OOP

- الفكرة الأساسية وراء لغات البرمجة غرضية التوجه هي دمج المعطيات والتوابع التي تعمل على تلك المعطيات في كينونة واحدة تسمى غرض (Object)، وعادة تزود توابع الغرض والتي تسمى توابع الأعضاء (Member functions)- الطريقة الوحيدة للوصول إلى المعطيات، لذلك تكون المعطيات محمية من التعديلات والخطأ ويقال أن المعطيات وتوابعها مغلفة (Encapsulated) في كينونة واحدة.
- تعتبر البرمجة غرضية التوجه طريقة قوية للتعامل مع المسائل البرمجية , فقد ظهرت في كل نقطة من مراحل تطورها طريقة جديدة تساعد المبرمجين على التعامل مع البرامج المتزايدة التعقيد .
- رغم أن البرمجة البنيوية كانت تعطي نتائج ممتازة عندما تطبق على برامج متوسطة التعقيد , إلا أنها كانت تفشل عند بعض النقاط حين يصل البرنامج إلى حد معين .

- تأخذ البرمجة OOP أفضل ميزات البرمجة البنيوية وتضم إليها مبادئ قوية وجديدة تسمح بتنظيم البرامج بفعالية أكبر .
- تعتمد البرمجة OOP على تحليل المشكلة إلى أجزائها الأساسية بحيث يصبح كل عنصر أو جزء عبارة عن غرض object محتوى ذاتياً يضم تعليماته الخاصة والمعطيات العائدة إليه . وأصبح المبرمج يستطيع إدارة برامج أكبر .

سمات البرمجة غرضية التوجه

- تشترك جميع لغات البرمجة OOP بما فيها C# بثلاث سمات مميزة مشتركة وهي :

1. التغليف Encapsulation (الصفوف Classes) .
2. الوراثة Inheritance .
3. تعدد الأشكال Polymorphism .

Encapsulation (Classes)التغليف

- يطلق مصطلح التغليف على عملية ربط البيانات ونص البرنـامج الذي يتعامل معها وحمايتها من التداخل وسوء الاستخدام الخارجيين .

- نطلق على البيانـات ونـص البرنـامج المـرتبطين بهذه الطريقـة اسم " الغرض Object " . فالغرض هو الآلية التـي تدعم التغليف فـي الـ OOP .

- يمكن أن تكون البيانات أو نص البرنامج أو كليهما خاصة بهذا الغرض أو عامة .

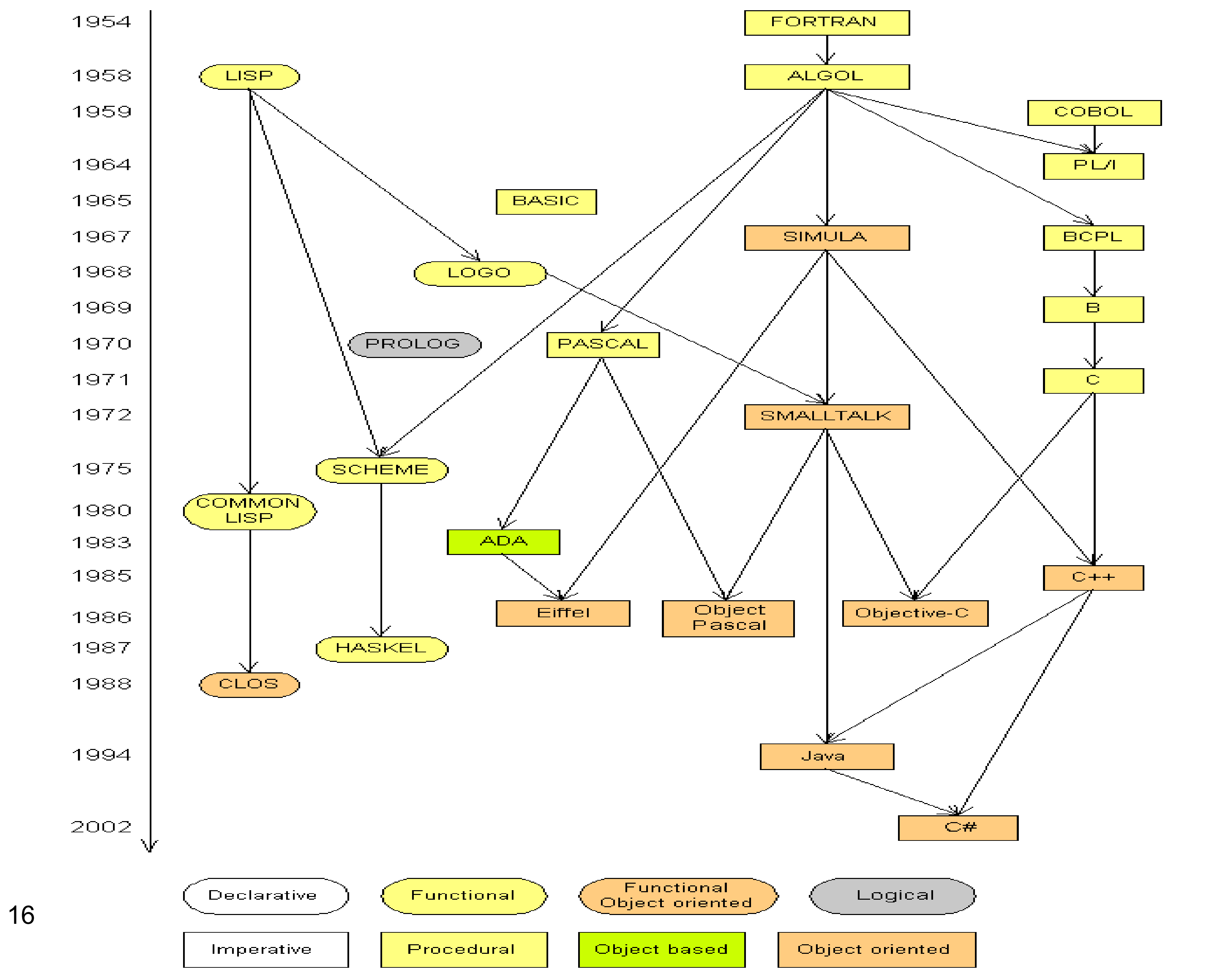
- يمكن الوصول إلى البيانات ونصوص البرامج الخاصة في الغرض فقط من قبل أجزاء البرنامج داخل الغرض أما عندما تكون البيانـات أو نصوص البرنـامج عامة يمكن الوصول إليها من أجزاء البرنـامج خارج الغرض .

inheritance الوراثة

- يطلق مصطلح الوراثة على العملية التي يكتسب غرض ما من خلالها على خصائص غرض آخر .
- أي يستطيع غرض ما أن يرث مجموعة عامة من الخصائص ثم يضيف إليها الميزات الخاصة به وحده .
- في الوراثة يستطيع صفاً ما أن يرث صفاً أساسياً وذلك بتحديد الصف الأساسي الذي ينتمي إليه بالإضافة إلى الصفات الخاصة التي تجعله مميزاً عن غيره .

polymorphism تعدد الأشكال

- تعدد الأشكال تمثل الميزة التي تسمح لاسم واحد أن يُستخدم لهدفين مترابطين أو أكثر لكنهما مختلفين تقنياً .
- أما لغة C# فتدعم تعدد الأشكال ويمكن استدعاء هذه التوابع بنفس الاسم ويحدد نوع البيانات المستخدم لاستدعاء التابع النسخة المحددة من التابع الواجب تنفيذها .
- تكمن فائدة تعدد الأشكال في أنها تساعد على تخفيف التعقيد بالسماح لواجهة واحدة أن تعرف بمجموعة عامة من الأفعال .
- تسمح طريقة تعدد الأشكال بالتعامل مع تعقيد أكبر من خلال إنشاء واجهة قياسية تشمل جميع الفعاليات المترابطة .



1954
1958
1959
1964
1965
1967
1968
1969
1970
1971
1972
1975
1980
1983
1985
1986
1987
1988
1994
2002
FORTRAN
ALGOL
LISP
COBOL
PL/I
BASIC
SIMULA
BCPL
LOGO
B
PROLOG
PASCAL
C
SMALLTALK
SCHEME
COMMON LISP
ADA
C++
Eiffel
Object Pascal
Objective-C
HASKEL
CLOS
Java
C#
Declarative
Functional
Functional Object oriented
Logical
Imperative
Procedural
Object based
Object oriented

بنية البرنامج في لغة C#

- بنية البرنامج

```
1. using System ;
2.  namespace   Example1 {
3. class    ClassName
4.     {
5.         static  void   Main( string args[ ] )
6.         {
6.              body {          Body of main( )
7.       } // end method main
8.    } // end class Name
9.   } //end namespace
```

مثال(1)

```
1.  //This program will display a message on the screen.
2.  using System ;
3.
4.  class     Welcome {
5.  static  void   Main( string [ ] args ) {
6.
6.    Console.WriteLine(" Welcome to C# Programming \n");
      Console.ReadKey();
7.      } // end method main
8.    } // end class Name
9.
```

RESULT

```
Welcome to C# Programming
Press any key to continue . . .
```

التعليقات
Comments

//This program will display a message on the screen.

- يبدأ هذا السطر من البرنامج بالشرطة المزدوجة (//) الدالة على أن بقية السطر عبارة عن تعليق (comment)، تضاف التعليقات إلى البرامج لتساعد المبرمج أو أي شخص آخر قد يحتاج إلى قراءة البرنامج على فهم ما الذي يفعله البرنامج، لذا من المستحسن أن يبدأ كل برنامج في لغة C# بتعليق يوضح الغرض الذي من أجله كتب البرنامج.
- تستخدم الشرطة المزدوجة (//) إذا كان التعليق يمتد لسطر واحد فقط single-line comment.
- هنالك نوع آخر من التعليقات يتيح لنا كتابة تعليقات تمتد إلى عدة أسطر multi-line comments ، نستطيع كتابة التعليق السابق كما يلي:

```
/*
    This program will display
    a message on the screen
*/
```

- يبدأ الرمز" */ " ثم التعليق وينهتي بالرمز " /*" . نجد أن نهاية السطر لا تعنى انتهاء التعليق لذا يمكننا كتابة ما نشاء من أسطر التعليقات قبل الانتهاء بالرمز " */ " .
- يمكن اختيار نوع التعليق الذي تراه مناسباً .
- **التابع ReadKey() يقوم بإيقاف الشاشة إلى حين الضغط على أي زر من لوحة المفاتيح .**

فضاءات الأسماء (namespace)

- تملك البرامج الضخمة عادة الكثير من الصفوف ، وهذا يعني أنه من الوارد جداً أن يكون لصفين الاسم نفسه ، عندها يصبح المترجم مشوشاً لا يعرف ما هو الصف الذي نتحدث عنه .
- لمنع المترجم من الالتباس نقوم بتنظيم الصفوف ضمن فضاءات أسماء .ولكتابة الاسم الرسمي للصف نقوم بإلحاق الصف باسم فضاء الأسماء الذي يتضمن هذا الصف ، فمثلاً : إذا كان اسم فضاء الأسماء في الصف Welcome هو **Example1** فإن الصف Welcome سيحمل رسمياً الاسم :

Example1.Welcome

- مما يجعله مميزاً عن الصف **Welcome** الموجود في إطار عمل NET. ، وهو فضاء الأسماء System .Drawing وبالتالي فإن الاسم الرسمي له هو :

System .Drawing.Welcome

- نستخدم في السطر 6 الصف Console ، وهو موجود في فضاء الأسماء System .

- ونحتاج في الحالة الافتراضية إلى الاسم الكامل للوصول إلى الصف Console أي إلى System.Console ، ولكن من المتعب أن نكتب (.System) في كل مكان ، لذلك يمكننا اختصار الأمر بأن نعلم المترجم أننا نستخدم فضاء الأسماء System في برنامجنا . وبذلك يمكننا أن نستخدم مباشرة أي صف ثم التصريح عنه ضمن فضاء الأسماء System بدون أن نكون مضطرين لوضع الاسم الرسمي كاملاً .
- توضع تعليمات **using** في أعلى البرنامج قبل أية تصريحات أو تعليمات أخرى كما هو مبين في السطر 2 من البرنامج (**using System**) .
- يمكن إنشاء فضاءات خاصة بنا ، ويتم ذلك باستخدام الكلمة المفتاحية **namespace** متبوعة باسم فضاء الأسماء الجديدة ، ثم زوج من الأقواس { } كما هو مبين في البرنامج في السطر 3 حيث فضاء الاسم الجيد هو **Example1** وكل صف يتم تعريفه ضمن القوسين يكون منتمياً إلى فضاء الأسماء الجديد هذا .
- يجب أن يتضمن كل برنامج #C صفاً واحداً على الأقل ، لقد قمنا في المثال السابق في السطر 4 بتعريف صف اسمه Welcome ويوضع جسم الصف ضمن قوسين أي من السطر 5 (فتح القوس) إلى السطر 8 (إغلاق القوس) .

التابع ( )Main
Main Method

- هناك دائماً في كل برنامج تابع اسمه Main ، وهو يعتبر نقطة البدء في البرنامج ، وهذا التابع مميز لأن CLR تستدعيه بعد تحميل البرنامج إلى الذاكرة , وبمجرد أن تقوم CLR باستدعاء التابع Main ينطلق تنفيذ البرنامج وعندما يعيد التابع Main ينتهي البرنامج كما هو مبين في البرنامج السابق .

```
1. public static  void   Main( string [ ] args[ ] )
2.       {

3.   Console.WriteLine(" Welcome to C# Programming \n");

4.      } // end method main
```

Common Language Runtime(CLR)
زمن تشغيل اللغة المشتركة

- **يمكن القول أن التابع Main** هو كتلة الشيفرة التي تعرف CLR وإن عليها استدعاءها عندما نبدأ بتنفيذ البرنامج ، وباعتبار أن على CLR استدعاءه ، **فمن الضروري أن نقوم بتعريف التابع Main ساكناً static ويجب أن يكون اسمه Main بحرف M كبير** , وليس صغير main كما هو الحال في لغة C++ ولغة Java ، لأن لغة C# تميز بين الأحرف الكبيرة والصغيرة .
- **وبما أن التابع يمكن أن يعيد بيانات** ، فمن الضروري أن يحدد كل تابع نوع البيانات التي سيعيدها ، فإذا كان التابع لا يعيد بيانات فمن الضروري تحديد نوع الإعادة void ، **وبما أن التابع Main في المثال السابق لا يعيد بيانات فإن نوع الإعادة هو void .**
- **يتضمن جسم التابع Main في المثال السابق سطراً واحداً** من الشيفرة وهو السطر 6 ، **ويتم فيه استحضار التابع Main من الصف Console .**
- **أما التابع WriteLine فيقوم بطباعة سلسلة المحارف التي بين علامتي الاقتباس المزدوجة على الشاشة** كما هي وهو موجود ضمن الصف Console **والانتقال إلى سطر جديد** ، أم إذا كتبنا التابع Write فيقوم بطباعة سلسلة المحارف دون الانتقال إلى سطر جديد .

- أما متوالية الهروب **\n** فهي تجبر المترجم إلى الانتقال إلى سطر جديد .
- **يجب أن تنتهي كل تعليمة في برنامج C# بفاصلة منقوطة** “ ; “ (semi colon).
- إذا كتبنا السطر 6 من البرنامج السابق كما يلي :

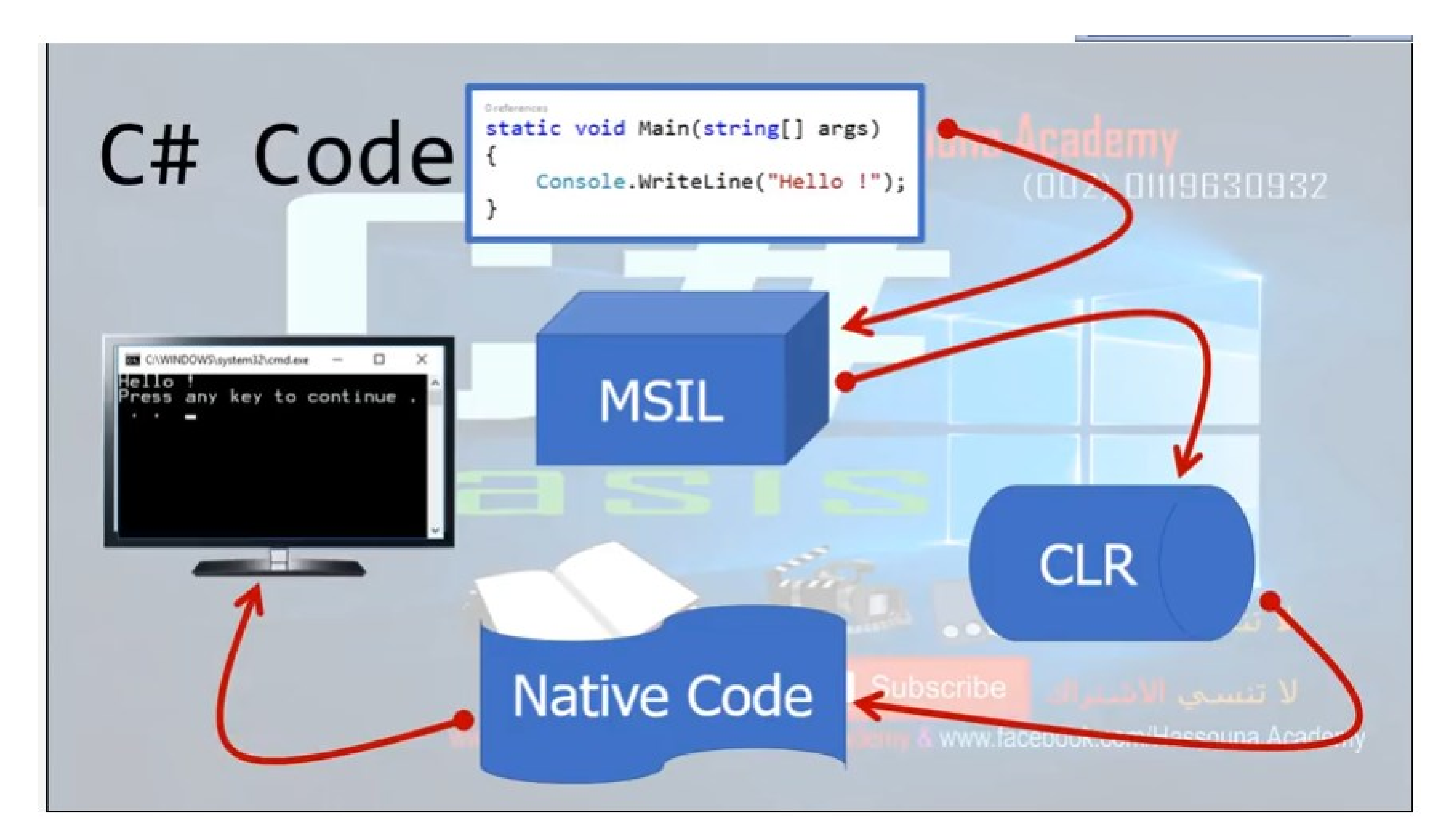
```
Console.WriteLine(" Welcome  \n to C# \n Programming \n");
```

عند تنفيذ البرنامج سوف نحصل على الخرج التالي :

RESULT

```
Welcome
to C#
Programming

Press any key to continue . . .
```



C# Code
static void Main(string[] args)
{
Console.WriteLine("Hello !");
}
MSIL
CLR
Native Code
C:\WINDOWS\system32\cmd.exe
Hello !
Press any key to continue . . .

توابع الإدخال والإخراج

- **هناك حاجة لوجود طريقة لإدخال المعطيات من المستخدم وإظهار النتائج على الشاشة** ، **ونحتاج أيضاً إلى تحديد تنسيق الخرج** ، وذلك باستخدام عرض للحقل ورموز التنسيق التي يمكن من خلالها عرض البيانات كعملة أو أية تنسيقات أخرى للحصول على واجهة تحقق أكثر تفاعلاً مع المستخدم .
- استخدمنا في المثال السابق التابع WriteLine الخاص بالصف Console لطباعة نص (سلسلة محرفية) text على الشاشة وقد قمنا بطباعة النص :

Welcome to C# Programming

- يقوم التابع WriteLine بطباعة النص والانتقال إلى سطر جديد .
- قد لا نرغب أحياناً بالانتقال إلى سطر جديد بعد طباعة النص وانما الاستمرار بالطباعة على السطر نفسه عندها نستخدم التابع Write.

تابع الإخراج Write

- من المفيد أحياناً أن تتم طباعة النص على الشاشة بدون الانتقال إلى سطر جديد ، لذلك نستخدم التابع Write من الصف Console لأداء هذه المهمة .

مثال(2)

```
1.  //This program will display a message on the screen.
2.  using System ;
3.
4.  class     Welcome
5.     {
6.        static  void   Main( string [ ] args )
7.        {
6.    Console.Write (" Welcome to  C#  Programming ");
      Console.ReadKey();
7.      } // end method main
8.    } // end class Name
```


•

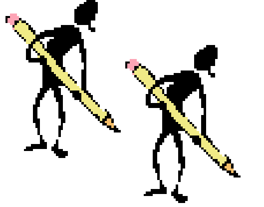
- يقوم البرنامج بطباعة النص **Welcome to C# Programming** وكذلك العبارة **Press any key to continue** على سطر واحد كما يلي :

RESULT

Welcome to C# Programming Press any key to continue

- يمكننا جعل التابع Write يتصرف كالتابع WriteLine وذلك بوضع محرف الانتقال إلى سطر جديد في نهاية السلسلة المحرفية ، حيث إن محرف الانتقال إلى سطر جديد (\n) هو محرف خاص يميزه المترجم عن غيره ويعني " انتقل إلى السطر التالي " ويصبح السطر 6 من البرنامج كما يلي :

```
6. Console.WriteLine(" Welcome to C# Programming \n");
```



مثال(3)

```
1.  /* Example2  :This program will display a character
    and integer number on the screen   */
2.  using System ;
3.  class     Program2
4.     {
5.  static  void   Main( string [ ] args )
6.         {
6.    Console.Write (7 + " is an integer.\n ") ;
7.    Console.Write ('a' + " is charachter.\n ") ;
7.       } // end method main
8.    } // end class Name
```

RESULT

```
7  is an integer.
a  is a character .
Press any key to continue
```

- **من خرج البرنامج يتضح لنا ما يلي:**

1- يتم حصر النص المطلوب ظهوره على الشاشة بين علامتي اقتباس مزدوجة " is an integer ".

2- تتم كتابة الثوابت الرقمية بدون علامتي اقتباس مثلاً العدد 7.

```
Console.Write (7 + " is an integer.\n ") ;
```

3- يتم حصر حرف واحد مطلوب ظهوره على الشاشة بعلامة اقتباس فردية المحرف 'a'.

```
Console.Write ('a' + " is charachter.\n ") ;
```

تقوم بعض اللغات كـ Basic مثلاً بالانتقال إلى سطر جديد تلقائياً في نهاية كل عبارة خرج ، لكن C# لا تفعل ذلك كما أن العبارات المختلفة والموضوعة في أسطر مختلفة لا تؤدي إلى ذلك .

متواليات الهروب (Escape Sequences)

- نلاحظ في المثال(2) أن السلسلة المحرفية في تعليمة Write تنتهي بـ \n وهي تعلم الصف Console أن ينتقل إلى السطر التالي، المحرف \n هو نوع خاص من المحارف التحكمية، و \ تسمى الشرطة الخلفية (Back slash) أو حرف هروب (Escape character) وتسمى هي والحرف الذي يليها متواليات الهروب.
- **متواليات الهروب \n** يعنى الانتقال إلى سطر جديد حيث يجبر المؤشر على الانتقال إلى بداية السطر التالي ،وهناك محارف تحكمية أخرى مثل محرف الجدولة ومحرف تمرير صفحة وغيرها، ونعرض بعض متواليات الهروب الشائعة في الجدول التالي .

- **متوالية الهروب** **الوصف**
- \n سطر جديد.
- \t مسافة أفقية(محرف الجدولة).
- \b محرف التراجع back space.
- \\ لطباعة شرطة خلفية.
- \r حرف الإرجاع، يجبر المؤشر على الانتقال إلى بداية هذا السطر.
- \ " لطباعة علامة اقتباس مزدوجة
- \’ لطباعة علامة اقتباس مفردة

- يمكن استخدام محرف الهروب ليس فقط من أجل إخبار المترجم أن عليه التعامل مع المحارف العادية على أنها محارف تحكم ، ولكنه يستخدم أيضاً لإخبار المترجم أن عليه التعامل مع محارف خاصة محرفية كمحارف عادية .
- مثلاً علامات الاقتباس المزدوجة ، تستخدم لبدء وإنهاء السلسلة المحرفية ، إذا وضعت شرطة خلفية أمام علامة الاقتباس المزدوجة " Hello \ " C# " فإن المترجم سيتعامل معها على أنها محرف عادي وستتم طباعته على الشاشة أي طباعة (") وليس كمحرف إنهاء السلسلة المحرفية . مثلاً :

```
Console.Write (" Hello \ " C# ") ;
```

هناك ثلاث مجموعات من علامات الاقتباس المزدوجة في التصريح عن السلسلة المحرفية ، تقوم الأولى والأخيرة كالمعتاد ببدء وإنهاء السلسلة المحرفية ، ولكن علامة الاقتباس المزدوجة في الوسط هي في الواقع جزء من السلسلة المحرفية ، لذلك علينا إخبار المترجم أن يتعامل معها كمحرف عادي لذلك وضعنا شرطة خلفية قبلها .

هنا سوف يتم طباعة النص (Hello " C#)

- **عندما نريد طباعة الشرطة الخلفية كشرطة خلفية .**
  - بما أن المترجم يستخدم الشرطة الخلفية كمحرف هروب فإننا بحاجة لطريقة نخبر بها المترجم أن يتعامل معها كمحرف عادي في الحالات التي نريد فيها فعلياً أن تكون الشرطة الخلفية جزءاً من السلسلة المحرفية .
  - تصادفنا هذه الحالة كثيراً عندما نقوم بتخزين أسماء الملفات كسلاسل محرفية . لأن المسارات إلى أسماء الملفات تتضمن شرطات خلفية لفصل أسماء المجلدات .
  - تستخدم شرطة خلفية مزدوجة لإخبار المترجم أن يتعامل مع الشرطة الخلفية كمحرف عادي **مثلاً : لتكن لدينا السلسلتين المحرفيتين التاليتين :**

```
string  s1 = " C: \\ test.txt " ;
string  s2 = " C: \  test.txt " ;
```

عند طباعة هاتين السلسلتين سنحصل على الخرج التالي :

```
C: \  test.txt
C:          est.txt
```

المشكلة بالنسبة للملف الثاني هو أن المترجم يتعامل مع الشرطة الخلفية كمحرف هروب ، وبالتالي يتم تحويل حرف t الذي يليها إلى محرف جدولة tab .

- **باختصار** ، علينا وضع محرف هروب تليه شرطة خلفية أخرى إذا أردنا أن تكون الشرطة الخلفية جزءاً من السلسلة المحرفية بدلاً من أن تستخدم كمحرف هروب .
- **عندما نرغب بتخزين اسم ملف ضمن متحول من النوع string** ، علينا استخدام شرطة خلفية مزدوجة بعد أسماء المجلدات ، وإلا فإن المترجم سيعتقد أننا نحاول استخدام محرف هروب .
- **قد تكون الشرطات الخلفية مصدر إزعاج** ، فقد قررت مايكروسوفت استخدام الرمز @ كإشارة إلى المترجم تقول بأن السلسلة هي سلسلة محرفية لا تتضمن أية محارف هروب ، وبالتالي يصبح المثال السابق كما يلي :

```
string  s1 = " C: \\ test.txt " ;
string  s2 = @" C: \  test.txt " ;
```

أي أن وجود المحرف @ في بداية السلسلة المحرفية تخبر المترجم أنه لا توجد أية محارف هروب في هذه السلسة ، وبالتالي يتم التعامل مع الشرطة الخلفية كجزء من السلسلة المحرفية .

تنسيق الخرج

- يمتلك التابعان Write و WriteLine ميزة سهلة الاستعمال تتيح لنا تمرير المتحولات كوسطاء إضافية يمكن تنسيقها في الخرج ، يبين المثال التالي كيف يتم الاستدعاء .




```
1. using System ;
2. class    Program3  {
3. static  void   Main( string [ ] args )  {
4. string   name = " Waseem  younes ";
5. double  temprature = 37.5;
6. Console.WriteLine(" Name {0} \n temperature {1} ",name ,temperature);
7. Console.WriteLine( "{0} \n {1}", "Welcome to", "C# Programming!" );
7.       } // end method main
8.    } // end class Name
```

- **تم التصريح في هذا المثال** عن متحول من النوع string اسمه name لتخزين الاسم ، ثم عن متحول من نوع double اسمه temprature وأسندنا له القيمة 37.5 .
- **قمنا بوضع محددي موضع (place holder) ضمن السلسلة في التابع WriteLine** ، حيث محدد الموضع الأول هو {0} وسيتم ملؤه بالمتحول الأول الذي يلي السلسلة المحرفية ، أما محدد الموضع الثاني فهو {1} وسيتم ملؤه بالمتحول الثاني الذي يلي السلسلة المحرفية ، ويظهر الخرج كما يلي :

```
RESULT
Name   Waseem  younes
temprature  37.5
Press any key to continue
```

- يمكن أن تضمن محددات الموضع عرض الحقول ورموز تنسيق لتحديد كيفية عرض المتحولات .
- يوضع عرض الحقل ضمن قوسين {...} ويفصل عن الرقم الأول بفاصلة ، ويبين المثال التالي كيف يتم تحديد عرض الحقل من أجل الخرج .

مثال(5)

```
1.  using System ;
2.  class    Program4  {
3.  static  void   Main( string [ ] args )  {
4.  int    x = 20 , y = 30  ;
5.  Console.WriteLine(" ( X , Y ) is ( {0,2} , {1,2} )", x, y);
7.       } // end method main
8.    } // end class Name
```

RESULT

(X , Y) is (20 , 30)

Press any key to continue

- لقد تمت طباعة المتحولين x و y وكل منهما يشغل حقلاً بعرض محرفين .
- يمكننا أيضاً استخدام رموز التنسيق لإخبار التابع WriteLine أننا نريد طباعة القيمة كعملة أو استخدام تنسيق خاص آخر ، ويبين البرنامج التالي رموز التنسيق الخاص بالعملة .




```
1. using System ;
2. class    Program5  {
3. static  void   Main( string [ ] args )  {
4. int    x = 20 , y = 30  ;

5. Console.WriteLine(" X  is  {0,7:C} \n Y is  {1,7:C} ", x, y);
6. } // end method main
7. } // end class Name
```

RESULT

X is $20 .00

Y is $30 .00

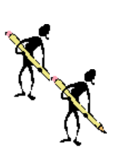
Press any key to continue

• ويبين الجدول التالي بعض رموز التنسيق التي تدعمها NET.

الخرج	الاستخدام	الوصف	الرمز
$ 200.00	{0,7:C}	عمله	C
50.00%	{0:P}	نسبة مئوية	P
1E+3	{0:E1}	أسي	E
44.2	{0:F1}	فاصلة عشرية	F
00340	{0:D5}	عشري	D
5.456E+05	{0:G4}	عام	G
86FD3	{0:X}	سداسي عشر	X

• تنسيق الخرج

```
string  date = string.format(“ {0}”,” 20/10/2017   12: 30 pm “);
Console.WriteLine( date) ;
```



تابع الإدخال

- يمكن أن يأتي الدخل من المستخدم باستخدام لوحة المفاتيح ، أو من ملف ، أو من مقبس Socket أو منفذ تسلسلي Serial Port أو من جهاز دخل آخر .
- سنستخدم الآن التابع ReadLine الموجود ضمن الصف Console والذي يقوم بإدخال سلسلة محرفية من المستخدم بواسطة لوحة المفاتيح .
- في المثال التالي التابع ReadLine يتيح للمستخدم إدخال اسمه ، ويقوم البرنامج بعدها ببناء رسالة مخصصة للمستخدم يعرضها على الشاشة باستخدام التابع WriteLine والذي يطلب منه إدخال اسمه .

مثال(7)

```
1.  using System ;
2.  class     Program6  {
3.  static  void   Main( string [ ] args )  {
4.  string    input  = “  “ ;
5.  Console.WriteLine(" Enter Your Name”);
6.   input= Console.ReadLine( );
7.   Console.WriteLine(" Hello , " + input + " Welcom to C#
                                             Programming ! ”);
7.       } // end method main
8.    } // end class Name
```

RESULT

```
Enter Your Name
Mamoun
 Hello , Mamoun Welcom to C# Programming !
Press any key to continue
```



اكتب برنامجا لإدخال رقمين صحيحين من المستخدم ويجمعهما و يطبع ناتج الجمع .

```
1.  using System ;
2.  class    Program7  {
3.  static  void   Main( string [ ] args )
4.  {
5.  int   integer1, integer2  ;
6.  Console.WriteLine("    Enter first Number\n ");
7.  integer1 = int.Parse( Console.ReadLine() );
8.  Console.WriteLine("  Enter   second Number\n ");
9.  integer2 = int.Parse( Console.ReadLine() );

10. int   sum = integer1 + integer2;
```

```
11. Console.WriteLine("  sum =  {0} " , sum);
12.  } // end method main
13.  } // end class Name
```

RESULT

Enter first Number
66
Enter   second Number
88
sum = 154
Press any key to continue

إدخال المعطيات المختلفة

- إذا كان المطلوب إدخال قيم حقيقية من قبل المستخدم نستخدم الصيغة التالية :

```
X = double.Parse( Console.ReadLine() );
```

حيث نقوم بتحويل السلسلة إلى قيمة حقيقية باستدعاء التابع Parse( ) .

- أما إذا كان المطلوب إدخال محرف واحد char من قبل المستخدم نستخدم الصيغة التالية :

```
ch = char.Parse( Console.ReadLine() );
```

- هناك طريقة أخرى لإدخال المعطيات المختلفة من المستخدم كما يلي :

```
Convert.ToInt(Console.ReadLine() )
```

- لقد قمنا بتحويل السلسلة المحرفية إلى قيمة رقمية بواسطة التابع ToInt( ) الموجود ضمن الصف Console .

- ويبين الشكل التالى كيفية التحويل إلى أنواع المعطيات المختلفة :

```
Example19.Arethmatic                          Main(string[] a
using System ;
namespace   Example19 {
class Arethmatic {
public static void Main(string   [ ]args ) {
 double   x , y ;
 char    ch ;
 Console.WriteLine("inter yuor x  &  y & ch ");
 x = double.Parse( Console.ReadLine() );
 y = double.Parse( Console.ReadLine() );
//ch = Convert.ToChar( Console.ReadLine())  ;
ch = char.Parse(Console.ReadLine());
   ch=Convert.
 Console.WriteL                          ---------- ");
   Console.Wri                           ') ;
 Console.WriteL                          ---------- ");
if( ch == '+')
 Console.WriteL
 //----------
 else if( ch =
 Console.WriteL
 //---------
   else  if(
```

ToBoolean
ToByte
ToChar
ToDateTime
ToDecimal
ToDouble
ToInt16
ToInt32
ToInt64

الأسئلة

- **حدد ما إذا كانت العبارات الآتية صحيحة أم خطأ:**
- التعليقات تجبر الحاسوب على طباعة النص الذي يلي // على الشاشة عند تنفيذ البرنامج.
- تتابع الهروب \n يجبر المؤشر على الانتقال إلى سطر جديد.
- برنامج #C والذي يقوم بطباعة ثلاث أسطر على الشاشة يجب أن يحتوى على ثلاث عبارات تستعمل WritLine.
- **ما هو الخرج من العبارة الآتية:**

```
Console.WriteLine(" \n **\n ***\n ****\n ") ;
```

رموز لغة C#

- يبنى البرنامج في لغة C# من مجموعة من العناصر التالية :
  - الحروف الانكليزية الصغيرة a, b , c, d, . . . x, y, z
  - الحروف الانكليزية الكبيرة A, B, C, D, . . . X, Y ,Z
  - الأرقام العربية 0, 1, 2, 2, . . . 8, 9
  - رموز خاصة مثل :

<   <=   >   >=   +   -   *   /   =   #   % $

?   ;   :   ,   .   <<   >>   !   ‘   ”

{ }   [ ] || &   ( )

العوامل الحسابية (Math Operator)

- لقد استعملنا عامل الجمع (+) لجمع integer1 و integer2، تتضمن C++ العوامل الحسابية التالية :
  - هي : - , + , % , / , *
  - الزيادة و النقصان : -- , ++
  - العمليات على البت Bitwise ~ , ^ , | , &
  - الازاحة Shift >>> , >> , <<
  - الاسناد =
  - والمعاملات التالية :
  - -= , += , *= , %= , /=

الجدول التالي يبين العوامل الحسابية الأساسية :

العامل	الوظيفة	التعبير الجبري	التعبيــر في C++
+	جمع	B+h	B+h
-	طرح	B-h	B-h
*	ضرب	Bh	B*h
/	قسمة	B/h	B/h
%	باقي القسمة	B mod h	B%h

- العوامل الأربعة الأولى تنجز أعمالاً مألوفة لدينا، أما عامل باقي القسمة % المسمى أيضاً المعامل modulus، يتم استعماله لحساب باقي القسمة لعدد صحيح على عدد آخر، لذلك فالتعبير 20%3 يساوى 2 .
- تسمى هذه العوامل الحسابية بالعوامل الثنائية لأنها تعمل على قيمتين.
- يمكن استعمال أكثر من عامل في تعبير رياضي واحد، فمثلاً التعبير:

```
C=(f-32)*5/9;
```

يحول درجة الحرارة من مئوية إلى فهرنهايت.(استعملت الأقواس لكي يتم تنفيذ الطرح أولاً بالرغم من أولويته المتدنية، يشير المصطلح أولوية Precedence إلى ترتيب تنفيذ العوامل، **العاملان * و / لهما أولوية أعلى من + و -).** وهذا ما سنراه لاحقاً بعد أن نتعرف على بقية عوامل C# .

العوامل العلائقية والمنطقية
Boolean/Relational Operators

- تقارن العوامل العلائقية قيمتين، وتؤدى إلى نتيجة صح/خطأ وفقاً لما إذا كانت المقارنة صح/خطأ.
- **هنالك ستة عوامل علائقية مبينة في كما يلي:**
- المساواة وعدم المساواة : != , ==
- علاقات : >= , <= , > , <
- علاقة الشرط (AND(Conditional : &&
- علاقة الشرط (OR(Conditional : ||

يبين الجدول التالية العوامل العلائقية أو المقارنة مع مثال لكل نوع

الرمز	المعنى	مثال
==	يساوى	a==b
!=	لا يساوى	a!=b
>	أكبر من	a>b
<	أصغر من	a<b
>=	أكبر من أو يساوى	a>=b
<=	أصغر من أو يساوى	a<=b

- تكون التعابير المبينة في عمود المثال صحيحة أو خطأ وفقا لقيم المتغيرين a و b.
- **فلنفرض مثلاً أن:**

a يساوى 9 و b يساوى 10.

التعبير : a == b خطأ.

- التعبير a!=b صحيح وكذلك التعبيرين a<b و a<=b ، والتعبيرين a>b و a>=b خطأ..

المعامل ثلاثي الحدود
The ? Operator

- يوجد في لغة جافا معامل خاص يسمح باختبار قيمة شرط منطقي واستخدام واحد من تعبيرين حسب قيمة هذا الشرط وله الصيغة التالية :

condition **?** expression – if – true **:** expression – if - else

مثال :

```
X = (y<5   ) ?  44 : 55 ;
X = (y<5  ?  44 : 55 ) ;
```

أولوية العمليات الحسابية

Operator	Type
( )	الأقواس
fabs( ),pow( ) , sqrt( ) ,sin( ),cos( )	التوابع
++ - -	Unary postfix
++ - - + -	Unary prefix
* / %	multiplicative
+ -	additive
< > >= <=	relational
== !=	equality
? :	conditional
= += -= *= /= %=	assignment




```
1.5 * 2.4 + 3.3 * 4.25 / 5.5

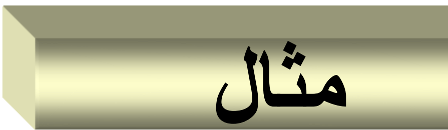
3.6       + 3.3 * 4.25 / 5.5

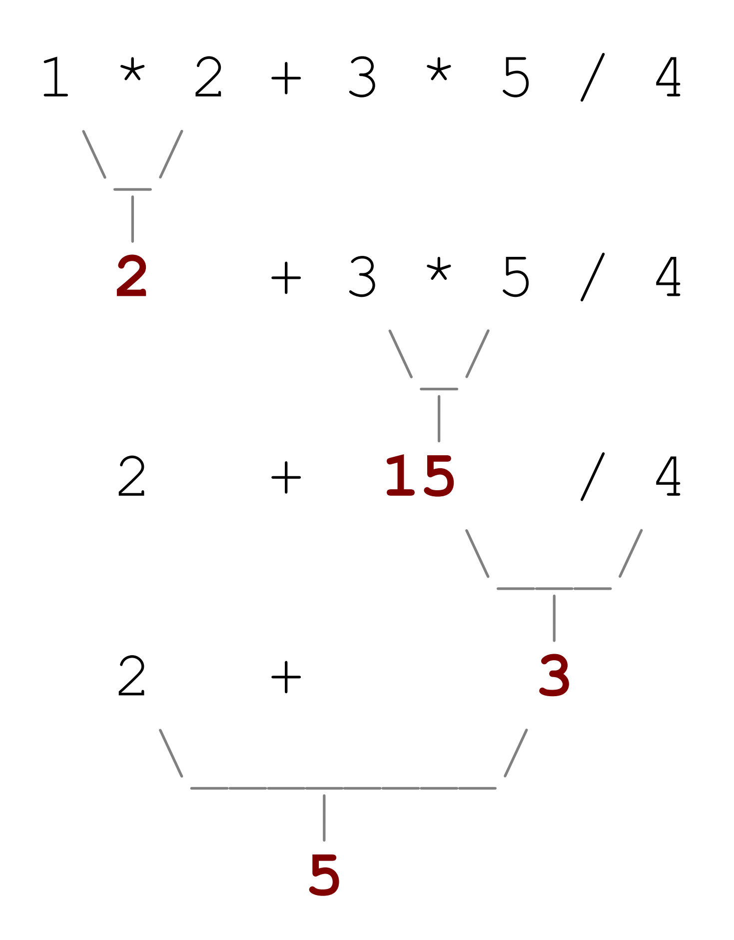
3.6       +   14.025   / 5.5

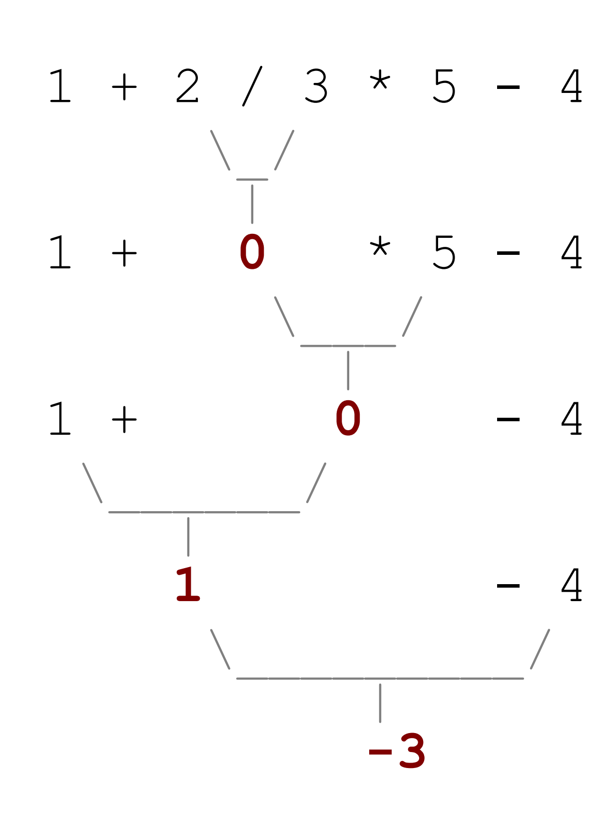
3.6       +          2.55

           6.15
```

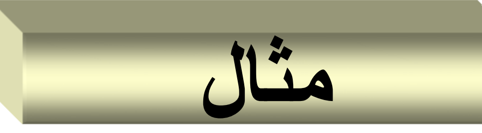
Precedence examples







Mixing integers and reals




```
7 / 3 * 1.2 + 3 / 2

2     * 1.2 + 3 / 2

2.4         + 3 / 2

2.4         +   1

      3.4
```

Precedence examples



- What values result from the following expressions?
  - `9 / 5`
  - `695 % 20`
  - `7 + 6 * 5`
  - `7 * 6 + 5`
  - `248 % 100 / 5`
  - `6 * 3 - 9 / 4`
  - `(5 - 7) * 4`
  - `6 + (18 % (17 - 12))`

- Which parentheses above are unnecessary (which do not change the order of evaluation?)

C# Keywords and contextual keywords

`if`	`implicit`	`in`	`int`	`interface`
`internal`	`is`	`lock`	`long`	`namespace`
`new`	`null`	`object`	`operator`	`out`
`override`	`params`	`private`	`protected`	`public`
`readonly`	`ref`	`return`	`sbyte`	`sealed`
`short`	`sizeof`	`stackalloc`	`static`	`string`
`struct`	`switch`	`this`	`throw`	`true`
`try`	`typeof`	`uint`	`ulong`	`unchecked`
`unsafe`	`ushort`	`using`	`virtual`	`void`
`volatile`	`while`			

Contextual Keywords

`add`	`alias`	`ascending`	`async`	`await`
`by`	`descending`	`dynamic`	`equals`	`from`
`get`	`global`	`group`	`into`	`join`
`let`	`on`	`orderby`	`partial`	`remove`
`select`	`set`	`value`	`var`	`where`
`yield`				

المتحولات والعوامل

- يستخدم كل برنامج أنواع مختلفة من المتحولات Variables ، ويستخدم كل برنامج المتحولات بطريقة ما .
- ففي برنامج Windows على سبيل المثال تكون جميع البيانات التي تمثل كل معالم النافذة ، كاللون والحجم والموضع والنص في شريط العنوان ، كل هذه البيانات مخزنة في متحولات .
- **المتحولات هي محددات مواقع في الذاكرة تخزن البيانات التي يستخدمها البرنامج .**
- هناك أنواع مختلفة من المتحولات المعرفة في لغة C# إضافة إلى إمكانية تعريف أنواع متحولات خاصة بالمبرمج .

التصريح عن المتحولات واستخدامها

- **قبل أن نستخدم المتحولات يجب التصريح عنها** ، فبهذه الطريقة يعلم المترجم ما هي العمليات التي يمكن الإجراء على هذه المتحولات وبالتالي يستدعي العوامل المناسبة لإنجاز هذه العمليات .
- **يتم التصريح عن المتحولات** من خلال تحديد نوع البيانات للمتحول بالإضافة إلى معرّف identifier وذلك حسب الصيغة التالية :

```
Varible _type     identifier ;
```

حيث :

Varible _type : هو نوع بيانات المتحول مثل int أو float أو غيرها .

Identifier : هو اسم المتحول الذي نريد استخدامه في البرنامج .

- **ينتهي كل تصريح عن متحول بفاصلة منقوطة** ، وعندما يصادف المترجم تصريحاً فإنه يطالب بمساحة له في الذاكرة بما يكفي لتخزينه .
- **أسماء المتحولات** : غالباً ما يبدؤون كتابة أسماء المتحولات بحرف صغير وكتابة حرف كبير عند بداية كل كلمة بعدها ، مثلاً : accountBalance .
- **لا يمكن في لغة C# أن** نستخدم متحولاً قبل أن نسند له قيمة ابتدائية ، حيث تمنع هذه القاعدة الكثير من الأخطاء البرمجية الشائعة في اللغات الأخرى .
- **يمكن أن نعطي المتحول قيمة ابتدائية** عند التصريح عنه أو نقوم بذلك لاحقاً ، ولكن يجب أن نعطيه قيمة ابتدائية قبل الوصول إليه .

قواعد كتابة المتحولات Variables

- **يجب إتباع ما يلي عند تسمية المتحولات :**

1. يجب أن يكون اسم المتحول معرفاً صحيحاً , أي سلسلة من الرموز تبدأ بحرف من حروف اللغة الانكليزية ن و لا يمكن أن يبدأ برقم .
2. يجب أن لا يكون كلمة محجوزة في لغة C# مثل for , أو أي قيمة منطقية false , true , او الكلمة null .
3. يجب أن يكون اسم المتحول فريداً في الكتلة التي عُرف بها .
4. يفضل أن يبدأ اسم المتحول بحرف صغير , وفي حال كان الاسم مؤلف من أكثر من كلمة يفضل جعل الحرف الأول من كل كلمة (ما عدا أول كلمة) حرفاً كبيراً مثل : isGoodColor
5. لا يمكن أن تتضمن المتحولات فراغات .

مجال المتحول

- مجال المتحول هو المنطقة التي يمكن فيها استخدام المتحول
- يحدد مجال المتحول متى يقوم النظام بإنشاء المتحول ومتى يقوم بتدميره من الذاكرة .
- **إن مكان توضع المتحول البرمجي داخل النص البرمجي هو الذي يحدد مجاله , وذلك حسب إحدى التصنيفات التالية :**

1- المتحولات الأعضاء Members Variable : هي معطيات أعضاء من صف , ويتم التصريح عنها داخل الصف ولكن خارج أي تابع أو بان , ومجال هذا النوع من المتحولات هو كامل الصف .

2- المتحولات المحلية Local Variables : يتم التصريح عنها داخل الكتلة , ومجالها يمتد من نقطة التصريح عنها ولغاية نهاية الكتلة التي صُرحت عنها فيها .

3- الوسطاء لتابع (Parameters function) : هي متحولات يتم استخدامها في التوابع أو البواني من أجل تمرير القيم من/إلى التابع أو الباني .إن مجال المتحولات هو كامل التابع .

- **يمكن تهيئة المتحولات الأعضاء والمتحولات المحلية مباشرة أثناء التصريح عنها :**

```
int x=200   ;   long y=300 ;
```

- الوسطاء لا يمكن اسناد قيم لها مباشرة , بل يجب تمرير قيمها من خلال النص البرمجي .



هذا المثال يعرض مجال المتحولات المحلية ضمن الكتل .

```
1.    // Demonstrate block scope.
2.  using System ;
3.  class Scope {
4.  static void Main(string [ ] args ) {
5.    int  x; // known to all code within main
6.    x = 10;
7.    if( x == 10) { // start new scope
8.    int  y = 20; // known only to this block
9.    // x and y both known here.
```

```
11. Console.WriteLine("x and y : {0} , {1}  " , x , y);
12.   x = y * 2;
13.   }//End if
14.    // y = 100; // Error! y not known here
15.   // x is still known here.
16.   Console.WriteLine("x is " + x);
17. } // end method main
18.  } // end class Name
```

RESULT

```
x and y: 10  20
x  is  40
Press any key to continue
```

- **المتحول x في السطر 6** يتم التصريح عنه ضمن مجال الـ main وبالتالي يكون معرفاً حتى نهاية الـ main وسهل الوصول إليه .
- **بينما المتحول y في السطر 9** فهو معرف ضمن كتلة الـ if فقط بينما المتحول x يكون معرفاً في هذه الكتلة مع المتحول y .
- **بينما في السطر 17** خارج الكتلة if يكون المتحول x معرفاً أما y فيكون غير معرفاً .

أنواع المتحولات

- **يعتبر نوع المتحول مهماً لعدة أسباب :**
  – الأول هو أن المتحول يحتاج أن يطلب من النظام **أنشاء مساحة في الذاكرة لتخزين المتحول** فيما لا يعرف ما هي المساحة التي سيطلبها بدون معرفة المتحول .
  – كما إن المترجم يحتاج **أن يعرف العمليات المقبول إنجازها** على متحول محدد بحيث يمكنه فرض القواعد الملائمة عند استخدام هذا المتحول ، فبدون معرفة النوع ليست لدى المترجم طريقة لمعرفة أي القواعد سيفرض .
- **هناك عدة أنواع أولية ضمن لغة C# .**
  – هناك أنواع رقمية لتمثيل الأعداد الصحيحة و الفاصلة العشرية .
  – هناك النوع char لتمثيل محرف واحد .
  – هناك النوع bool الممثل لقيمة بوليانية و هي إما صحيح true أو خطأ false
  – أما النوع string فيمثل سلسلة من المحارف .
  – يعرض الجدول التالي بعض أنواع المتحولات الشائعة في C# .

أمثلة عن القيم المخزنة	يستعمل لتخزين	اسم النوع
char ch= ‘ a’	أحرف(محارف) 2byte	**char**
short x= 222	أعداد صحيحة قصيرة مع إشارة 2byte	**short**
int z = 14543	أعداد صحيحة عادية مع إشارة 4byte	**int**
long y= 76543987	أعداد صحيحة طويلة مع إشارة 8byte	**long**
9176 321 236.01202,6	تمثيل قيمة بولانية true أو false 1byte	**bool**

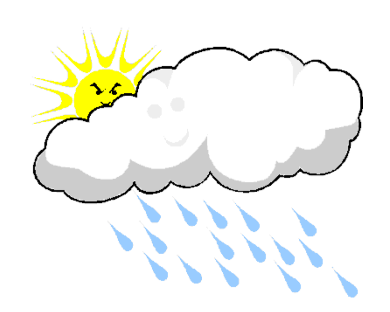
اسم النوع	يستعمل لتخزين	أمثلة عن القيم المخزنة
ushort	أعداد صحيحة قصيرة بدون إشارة 2byte	ushort a=222
uint	أعداد صحيحة عادية بدون إشارة 4byte	uint b=567432
ulong	أعداد صحيحة طويلة بدون إشارة 8byte	ulong c=654438765
float	أعداد حقيقية قصيرة مع إشارة 4byte	float n= 55.8
double	أعداد حقيقية مزدوجة مع إشارة 4byte	double k = 5437.8765
decimal	فاصلة عشرية مع 28 خانة إلى يمين الفاصلة .	decimal PI = 3.14159 ;

- تمتلك لغة جافا نوعين من أنماط المعطيات :
  - الأنماط الأساسية **Primitive Type**
  - الأنماط المرجعية **References Type**
- تستطيع الأنماط الأساسية تخزين قيمة واحدة فقط (رقم , محرف , قيمة منطقية , ...) وتحتاج إلى حجم معين في الذاكرة .
- **أنواع الأنماط الأساسية هي :**

Bool , byte, char, short,

int, long, float, double

- **الأنماط المرجعية هي :** الصفوف Classes والمصفوفات Arrays .



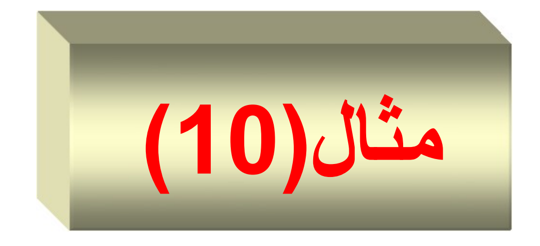
النمط boolean &cast

- النمط bool : هو 1Bit من المعطيات , ويأخذ قيمتين true أو false أي (1 , 0) .

Declaration: boolean b;

Assignment: b = true; b = false;

- المعامل (cast) يستخدم للتحويل من نوع إلى آخر من المعطيات .




```
// Demonstrate boolean values.
using System ;

class BoolTest  {
static  void   Main( string  [ ]args )  {
bool   b;
         b = false;
        Console.WriteLine("b is " + b);
        b = true;
        Console.WriteLine("b is " + b);
// a boolean value can control the if statement
         if(b) Console.WriteLine("This is executed." );
```

```
b = false;
if(b) Console.WriteLine("This is executed." );
// outcome of a relational operator is a boolean value
Console.WriteLine("10 > 9 is " + (10 > 9) );
} // end method main
  } // end class Name
```

RESULT

b is False
b is True
This is executed.
10 > 9 is True
Press any key to continue

byte , char & string المتحولات

- النمط byte : يُستخدم للأعداد الموجبة فقط 1byte=8bit
- النمط sbyte : يُستخدم للأعداد الموجبة الصحيحة والسالبة بحجم واحد بايت
- النمط char : هو محرف وحيد يتكون من 2bytes
  – يُستخدم لتخزين أي محرف من لوحة المفاتيح ويتم التعريف عن متحولات من النوع char, كما يلي :

char ch;

  – تنشئ مساحة في الذاكرة لمحرف وتسميه ch وحجمه 2Byte
  – ولتخزين محرف ما في هذا المتغير نكتب :

ch = ’z’

  – ودائماً تكون المحارف الثابتة كـ ‘a’ و’b’ محصورة بعلامة اقتباس فردية.
- النمط string يُستخدم لتعريف متحول يتكون من سلسلة محرفية .
- البرنامج التالي يعرض هذا النوع من المتحولات .

مثال(11)

```
using System ;

class BoolTest  {
static  void   Main( string [ ]args )  {
        char ch1 = 'X';
        Console.WriteLine("ch1 contains " + ch1);
        ch1++; // increment ch1
        Console.WriteLine("ch1 contains " + ch1);
} // end method main
  } // end class Name
```

RESULT

```
ch1 contains X
ch1 contains Y
Press any key to continue
```

المتحولات الصحيحة

- تمثل المتحولات الصحيحة أرقاماً كاملة أي قيم يمكن تعدادها ، كعدد أشخاص أو أيام أو عدد صفحات مثلاً ، ولا يمكن أن تكون الأعداد الصحيحة أرقاماً ذات نقطة عشرية ولكنها يمكن أن تكون سالبة.
- **هنالك ثلاثة أنواع من الأعداد الصحيحة في C#:**
  - short قصير
  - int عدد صحيح
  - long طويل
- وهي تحتل مساحات مختلفة في الذاكرة. الجدول التالي يبين هذه الأنواع والمساحة التي تأخذها في الذاكرة .
- **المثال التالي يحسب سرعة الضوء بالاعتماد على المتحولات الطويلة long .**

اسم النوع	الحجم	النطاق
Short	2byte	−32,768 إلـــــــى 32,767
int	4byte	-2147483648 إلــــــــــــــى 2147483647
long	8byte	2,147,483,648 إلــــــــــــــى 2,147,483,647




```
using System ;

class Light  {
static  void   Main( string [ ]args )  {
    long lightspeed, days, seconds, distance;
// approximate speed of light in miles per second
 lightspeed = 186000;//( m/s )
days = 1000; // specify number of days here
seconds = days * 24 * 60 * 60; //(second )
// compute //distance
distance = lightspeed * seconds; //(miles )
```

```
Console.Write ("In  " + days);
Console.Write ("days light will travel about " );
Console.WriteLine(distance + " miles.");
} // end method main
  } // end class Name
```

RESULT

In **1000** days light will travel about **16070400000000** miles.

Press any key to continue

المتحولات الحقيقية (float, double)

- يتم استعمال المتحولات الحقيقية لتمثيل قيم يمكن قياسها كالأطوال أو الأوزان.
- ويتم تمثيل الأعداد الحقيقية عادة برقم كامل على اليسار مع نقطة عشرية وكسر على اليمين.
- **هنالك ثلاثة أنواع من الأعداد الحقيقية** في أنظمة التشغيل الشائعة الاستعمال,وأشهر أنواع الأعداد الحقيقية من النوع double والذي يتم استعماله لمعظم توابع C# الرياضية,ويتطلب النوع float ذاكرة أقل من النوع double.
- و يوضح الجدول التالي هذه الأنواع والحجم الذي تأخذه في الذاكرة.

الحجم	اسم النوع
4byte	float
8byte	double
10byte	long double

المثال التالي يقوم بحساب مساحة الدائرة

مثال(13)

```
using System ;

class Light  {
static  void   Main( string [ ]args )  {
    double  Pi,  r,  a ;
    r = 10.8;// radius of circle
    Pi = 3.1416;// pi, approximately
    a = Pi * r * r;// compute area
    Console.WriteLine("Area of circle is  {0} " , a);
} // end method main
  } // end class Name
```

RESULT

Area of circle is 366.436224
Press any key to continue . . .

النوع object

- يمكن أن يكون أي نوع من أنواع المتحولات السابقة أي يمكن أن يكون :

int , float , double , char , decimal , string ……

- **مثال ذلك :**

```
object  x=313.22222m;
object  x=313.22222f;
object  x = 'a'
object  x = "Hello"
```

المتحولات الثابتة
Constant Variables

- يمكن التصريح عن المتحول بحيث تصبح قيمته ثابتة (أي غير قابلة للتعديل) **باستخدام الكلمة المحجوزة const** كما يلي :

```
const  int  x = 10  ;
```

- لا يمكن تغيير قيمة المتحول x إطلاقاً.

The + String Operator

- Strings in C# are objects (just like any other object), except that the language includes special support:
  - String literals are part of the language.
  - The string concatenation operator **+** is part of the language.

```
Console.WriteLine("Hello " + "World");
```

مثال على العمليات الحسابية

مثال(14)

```
// Demonstrate the basic arithmetic operators.
using System ;

class BasicMath {
public static void Main(string [ ]args) {
// arithmetic using integers
Console.WriteLine("Integer Arithmetic");
int  a = 1 + 1;
int  b = a * 3;
int  c = b / 4;
int  d = c - a;
int  e = -d;
```

```
Console.WriteLine("a = {0} " , a);
Console.WriteLine("b = {0} " , b);
Console.WriteLine("c = {0} " , c);
Console.WriteLine("d = {0} " , d);
Console.WriteLine("e = {0} " , e);
// arithmetic using doubles
Console.WriteLine("\nFloating Point Arithmetic");
double  da = 1 + 1;
double  db = da * 3;
double  dc = db / 4;
double  dd = dc - a;
double  de = -dd;
```

```
Console.WriteLine("da = {0} " , da);
Console.WriteLine("db = {0} " , db);
Console.WriteLine("dc = {0} " , dc);
Console.WriteLine("dd = {0} " , dd);
Console.WriteLine("de = {0} " , de);
// Demonstrate the % operator.
int x = 42;
double y = 42.25;
Console.WriteLine("x mod 10 = " + x % 10);
Console.WriteLine("y mod 10 = " + y % 10);
} // end method main
  } // end class Name
```

RESULT

Integer Arithmetic

a = 2

b = 6

c = 1

d = -1

e = 1

Modulus Operator

x mod 10 = 2

y mod 10 = 2.25

Floating Point Arithmetic

da = 2.0

db = 6.0

dc = 1.5

dd = -0.5

de = 0.5

Press any key to continue

اكتب برنامجا لإيجاد جذور المعادلة من الدرجة الثانية :

$$a.x^2 + b.x + c = 0$$

مثال(15)

```
using System ;

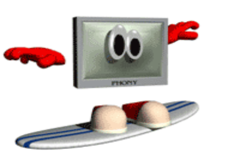
class Equation {
public static void Main(string [ ]args ) {
   double x1, x2, a, b, c, Delta;
Console.WriteLine(" Enter constants of your equation as:\n a b c = ");

   a = double.Parse( Console.ReadLine() );
  b = double.Parse( Console.ReadLine() );
  c = double.Parse( Console.ReadLine() );
   Delta = b*b - 4*a*c;
```

```
if(Delta >= 0)  {
        x1 = (-b +Math.Sqrt(Delta))/(2*a);
        x2 = (-b –Math.Sqrt(Delta))/(2*a);
 Console.WriteLine("x1 = " + x1 + "\t   x2 = " + x2 );
  }
  else
 Console.WriteLine("There are no real solutions of it!");
} // end method main
 } // end class Name
```

RESULT

```
Enter constants of your equation as:
 a b c =  1  5   6
x1 = -2       x2 = -3
Press any key to continue
```



الزيادة (++) و النقصان (- -)
Increment and Decrement

- الزيادة بمقدار واحد (++)

مثال

```
int A ;
A++ ; // A = A + 1 ;
Console.WriteLine(A );
```

- النقصان بمقدار واحد (--)

مثال

```
int  B ;
B- - ; // B= B – 1 ;
Console.WriteLine( B );
```




```
using System ;

class Increment {
public static void Main(string [ ]args ) {
int k=4;
++k ;
k++ ;
Console.WriteLine("k="+ k );
int i = k++;
Console.WriteLine("i="+ i + "  k = " + k );

int j = ++k;
Console.WriteLine("j="+ j + "  k = " + k );
  } // end method main
  } // end class Name
```

RESULT

```
6

6  7

8  8

Press any key to continue
```




```
// char variables behave like integers.
using System ;

class CharDemo2 {
public static void Main(String  [ ]args) {
char ch1 = 'X', ch2 = 'C';
Console.WriteLine("ch1 contains " + ch1+"\nch2 contains" +ch2);
ch1++; // increment ch1
ch2 - - ; // decrement ch2
Console.WriteLine("ch1 is now " + ch1+"\nch2 is now "  +ch2);
} // end method main
  } // end class Name
```

RESULT

```
ch1 contains  X
ch2 contains   C
ch1 is now      Y
ch2 is now     B
Press any key to continue
```

- يوضـح البرنــامج عمليــات الزيــادة increment (++) والنقصان decrement (- -) .
- نمـط المعطيـات المطبق هـو char أي محـرف واحـد ولـيس سلسلة محرفية .
- النتائج مبينة في الشكل السابق .

الأسئلة

- **أكتب عبارة #C صحيحة تقوم بالآتي:**
  - تعريف المتغيرات x ،y ، z و result لتكون من النوع int.
  - الطلب من المستخدم إدخال ثلاثة أرقام صحيحة.
- **حدد ما إذا كانت العبارات الآتية صحيحة أم خطأ:**
  - يجب الإعلان عن المتغيرات قبل استعمالها في البرنامج.
  - يجب تحديد نوع المتغيرات عند الإعلان عنها.
  - لا تفرق #C بين المتغيرات Number و number .

- اكتب برنامجاً يطبع على الشاشة أي معلومات شخصية عنك تريدها كـ (اسمك ، رقمك الجامعي ، تاريخ ميلادك ، عنوانك ....) بشكلٍ مُنسق ؛ مستخدماً تتابعات الهروب (\n) وَ (\t).
- اكتب برنامجاً يقوم بحساب مربع عدد حقيقي يُدخله المستخدم من لوحة المفاتيح ؛ ثم يطبع النتيجة على الشاشة .
- اكتب برنامجاً لتنفيذ العبارة الرياضية التالية ، بصيغ مختلفة باستخدام الأقواس.

```
y = 6+12/6*2-1;
y = (6+12)/(6*2-1);
y = (6+12/6)*2-1;
```

-

بنى التحكم

Control Structures

Chapter 2

محتويات الفصل الثاني

❑ بنى التحكم

- التعليمات الشرطية
  - ➢ تعليمة if
  - ➢ تعليمة if / else
  - ➢ تعليمة switch
- الحلقات
  - ➢ حلقة for
  - ➢ حلقة while
  - ➢ حلقة do / while



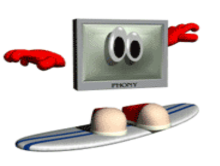


❑ تعليمات التفريع

- تعليمة break
- تعليمة continue
- تعليمة return

بنى التحكم Control Structures

- إن عدم استخدام بنى التحكم يؤدي إلى جعل مترجم لغة C# يقوم بتنفيذ تعليمات البرنامج بنفس الترتيب الذي ظهر في ملف المصدر .
- يمكن استخدام بنى التحكم لتغيير مسار تنفيذ البرنامج ، إما بوضع تعليمات شرطية (تنفذ عند تحقيق شرط معين فقط) أو حلقات تكرارية (تكرار مجموعة من التعليمات عدداً من المرات) .
- **تزودنا لغة C# بالعديد من تعليمات التحكم وهي :**

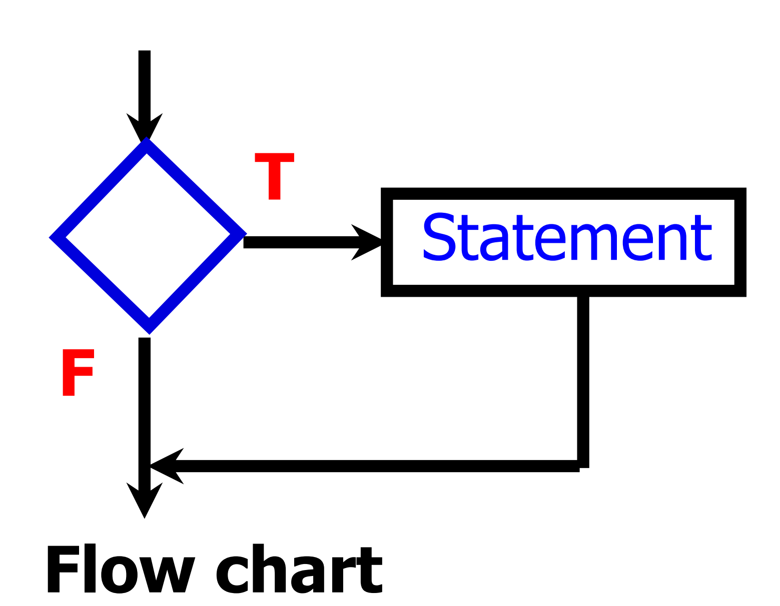


1- التعليمات الشرطية
التعليمة if

- تُستخدم التعليمة if لتنفيذ مجموعة من التعليمات عند تحقيق شرط ما وتنفيذ مجموعة تعليمات أخرى عند عدم تحقق الشرط .
- الصيغة العامة لتعليمة if ، هي :

```
if ( x < 0 )
x =  x * i;
```

```
if ( <condition> )
    {
        <statement>;
    }
```



Flow chart

If *expression* is true, *statement* is executed; otherwise *statement* is skipped.

Example:

1. **if** (grade >= 90)
2. **Console**.WriteLine(" Congratulations!\n”);
3. **Console**.WriteLine(" Your grade is ” + grade);

```
if ( condition )
  statement;

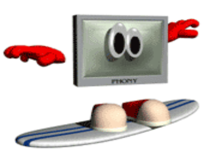
if ( condition ) {
  statement1;
  statement2;
  …
}
```

example code

```
if ( a < 12 )
   b = 45;

if ( x.length()<10) {
   x = x + "BLAH";
   y=x.length()-3;
}
```

- إذا كانت grade >= 90 فتُطبع العبارة التالية !Congratulations ثم يتم طباعة العبارة Your grade is 77 على سبيل المثال .
- أما إذا كانت أصغر من 60 فإنه يتم تجاوز السطر 2 ويتم تنفيذ السطر 3 فقط .
- يمكن ان يكون هناك أكثر من تعليمة if ويتم تنفيذها حسب ورودها ، بحيث التعليمة الأولى تنفذ أولاً ثم الثانية ... وهكذا .
- ويمكن ان تكون متداخلة ، بحيث يتم تنفيذ وإنهاء الـ if الداخلية وبعد ذلك يتم تنفيذ وإنهاء الـ if الخارجية ، كما هو مبين في الشكل التالي :



التعليمة if المركبة

The Compound Statement

Syntax

```
if (expression)
{
    statement;
    statement;
    if (expression)
    {
        statement;
        statement;
    }
}
```

Example:

```
if (u > v)
{
    a = 1;
    b = 2;
    if ( u > z)
    {
        x =11;
        y = 12;
    }
}
```

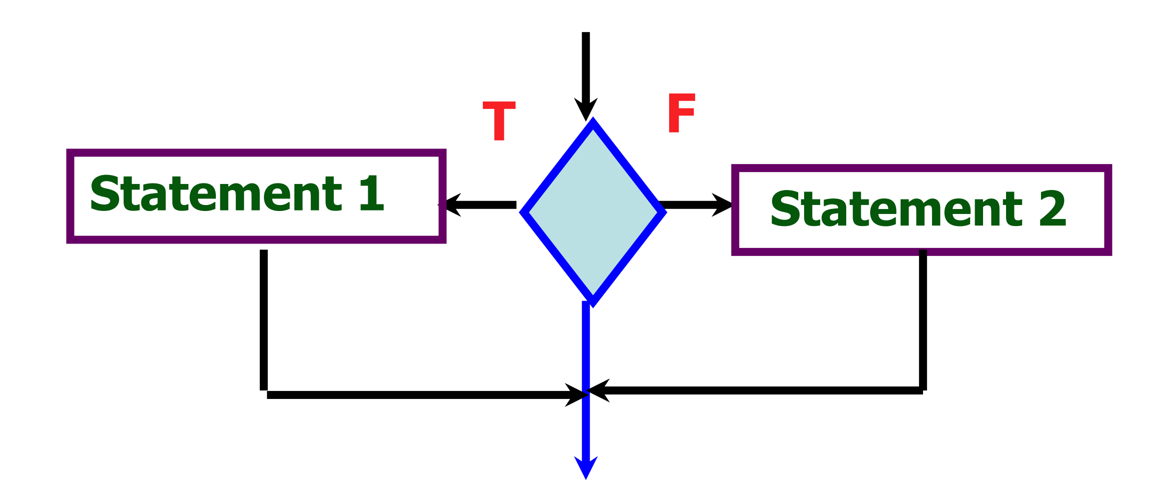
The compound statement is itself a statement.



التعليمة if /else

- عند استخدام التعليمة else مع التعليمة if فإنه يتم تنفيذ كتلة العبارة الأولى إذا تحقق الشرط true ، وإذا لم يتحقق الشرط فإنه يتم تنفيذ كتلة التعليمة else .
- الصيغة العامة :

```
if ( condition )
    statement;
else
    statement ;
```





المثال التالي يوضع التعليمة if / else والتعليمة if . حيث يتم إدخال رقم الشهر وطباعة الفصل الذي ينتمي إليه هذا الشهر .

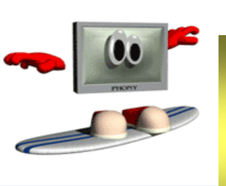
```
// Demonstrate if-else-if statements.
using System ;
class IfElse {
public static void Main(string [ ] args ) {
int month = 4; // April
String  season;
if(month == 12 || month == 1 || month == 2)
season = "Winter";
else if(month == 3 || month == 4 || month == 5)
season = "Spring";
```

```
else if (month == 6 || month == 7 || month == 8)
season = "Summer";
else if (month == 9 || month == 10 || month == 11)
season = "Autumn";
else
season = "Bogus Month";
Console.WriteLine("April is in the " + season + ".");
} // end method main
  } // end class Name
```

RESULT

April is in the Spring.

Press any key to continue



مثال(19)

المثال التالي يختبر درجة الطالب هل هي : A,B,C,D,F

```
//calculate the mark of a student
using System ;

class IfElse1 {
public static void Main(string [ ] args ) {
   double grade;
    Console.WriteLine("Enter  a  grade :");
    grade = double.Parse( Console.ReadLine() );
   if (grade >= 90)
 Console.WriteLine(" The mark is A\n ");
```

```
else if (grade >= 80 )
    Console.WriteLine(" The mark is B\n ");
 else if (grade >= 70 )
   Console.WriteLine(" The mark is C\n ");
 else if (grade >= 60  )
   Console.WriteLine(" The mark is D\n ");
 else if (grade < 60) {
    Console.WriteLine(" The mark is F\n");
   Console.WriteLine(" You must take this course again.\n");
   }
  } // end method main
} // end class Name
```





RESULT

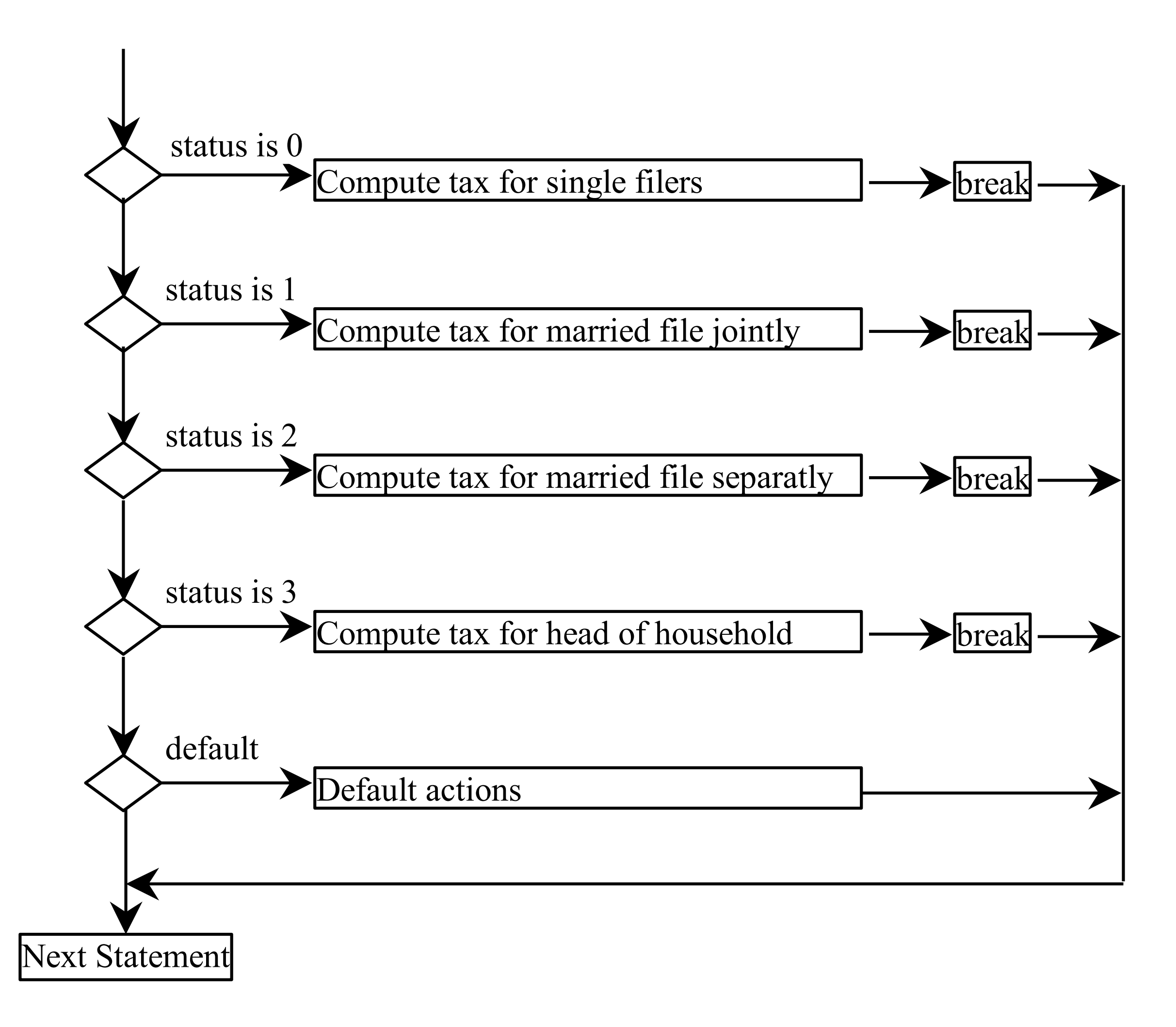
Enter a grade :
55
The mark is F
You must take this course again.

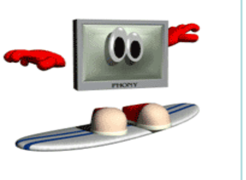
Press any key to continue

التعليمة switch
Switch (Multiple Decision

- التعليمة switch هي تعليمة تتحكم بتدفق البرنـامج ، **تبدأ بتعبير expression** وتنقل التحكم إلى إحدى **عبارات case** بناءً على قيمة ذلك المتغير .
- تعمل التعليمة switch مع الأنماط الأولية: int و short و byte و char وأنماط التعداد enumeration .
- تُستخدم عبارة break من أجل الخروج من العبارة switch
- إذا لم تحوي عبارة case الكلمة break فإنه سيتم تنفيذ السطر الموجود بعد نهاية العبارة case .
- يستمر التنفيذ إلى أن يتم الوصول إلى عبارة break أو الوصول إلى نهاية عبارة switch .

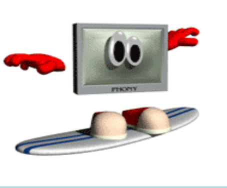
- يُسمح بعبارة default واحدة فقط في تعليمة switch وهي اختيارية وسيتم تنفيذها إذا لم تكن أي من الحالات الأخرى مقبولة
- **الصيغة العامة :**





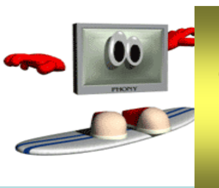
الصيغة العامة
Switch (Multiple Decision)

```
switch ( int expression ) {
   case int expression :
     statement;
     break;
   case int expression :
     statement;
     break;
   default :
     statement;
    break;
}//End switch
```



switch example

```
switch(food) {
   case 1:
      Console.WriteLine("Chicken");
      break;
   case 2:
      Console.WriteLine("Pizza");
      break;
   default:
      Console.WriteLine("Sorry, we are out");
     break;
}
```



مثال(20)

المثال التالي يوضح التعليمة Switch. حيث يتم ادخال رقم الشهر وطباعة الفصل الذي ينتمي إليه هذا الشهر.

```
// An improved version of the season program.
using System ;
class Switch {
public static void Main(string [ ] args ) {
int month = 4;
string season;
switch (month) {
case 12:
case 1:
case 2:
season = "Winter";
break;
```

```
case 3:
case 4:
case 5:
season = "Spring";
break;
case 6:
case 7:
case 8:
season = "Summer";
break;
```

```
case 9:
case 10:
case 11:
season = "Autumn";
break;
default:
season = "Bogus Month";
break;
}
Console.WriteLine("April is in the " + season + ".");
  } // end method main
} // end class Name
```

RESULT

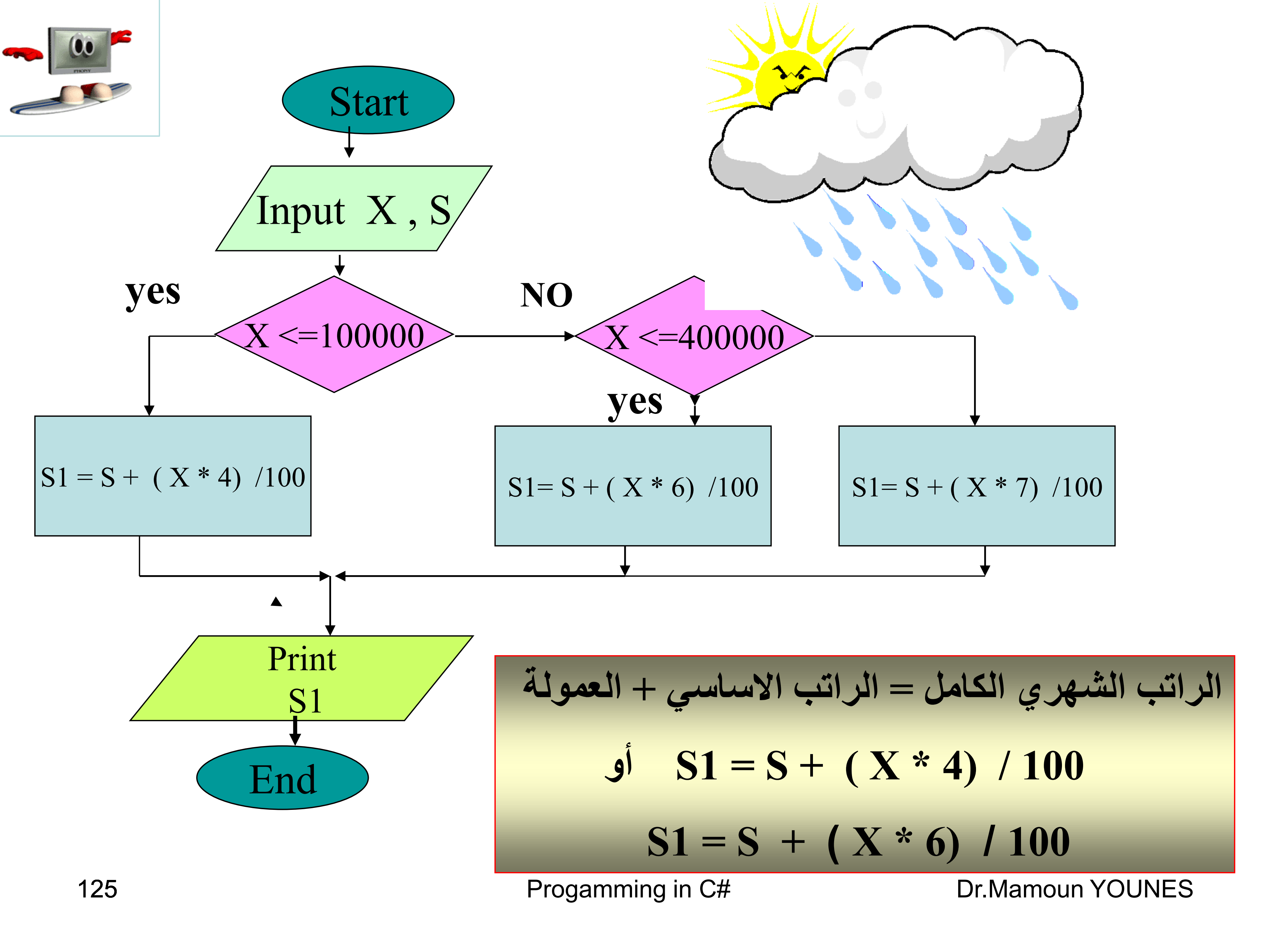
April is in the Spring.

Press any key to continue



مثال(21)

- مندوب مبيعات يتقاضى راتباً شهرياً أساسيا مقداره (1000) دولار ، ويتقاضى عمولة قدرها 4% إذا كانت المبيعات الشهرية لا تتجاوز (100000) دولار، ويتقاضى عمولة قدرها 6% إذا كانت المبيعات الشهرية لا تتجاوز (400000 دولار، ويتقاضى عمولة قدرها 7% إذا كانت المبيعات الشهرية أكثر من ذلك . والمطلوب :
  - اكتب برنامجاً بلغة C# يقوم بما يلي :
  - إدخال الراتب الشهري وقيمة المبيعات الشهرية وطباعتها
- حساب وطباعة الدخل الشهري الكامل للموظف .
- ارسم المخطط التدفقي المناسب .
- **<u>ملاحظة :</u> افترض S الراتب الشهري الأساسي ، X المبيعات الشهرية ، W العمولة ، S1 الراتب الشهري الكامل .**

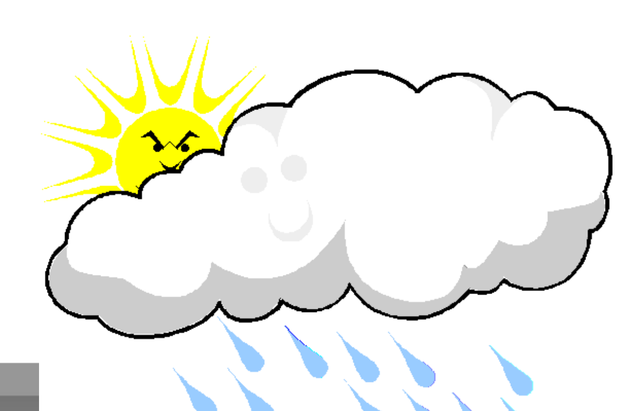

Start
Input X , S
yes
X <=100000
NO
X <=400000
yes
S1 = S + (X * 4) /100
S1= S + (X * 6) /100
S1= S + (X * 7) /100
Print
S1
End
الراتب الشهري الكامل = الراتب الاساسي + العمولة
S1 = S + (X * 4) / 100 أو
S1 = S + (X * 6) / 100



```
using System ;

class Ssalary {
public static void Main(string [ ] args ) {
  int  S   , X ;
  double  W ,S1 ;
 Console.WriteLine("inter yuor salary S & amount X ");
    S = int.Parse( Console.ReadLine()  );
   X = int.Parse( Console.ReadLine()  );
Console.WriteLine("-------------------------------");
```

```
if( X <= 100000 )
    W = X*4/100;
else if( X <= 400000 )
    W = X*6/100;
else
    W = X*7/100;
Console.WriteLine( " W= "  + W);
Console.WriteLine("-------------------------------");
S1 = S + W ;
Console.WriteLine("Allsalary =  " + S1);
Console.WriteLine("-------------------------------");
} // end method main
} // end class Name
```



RESULT

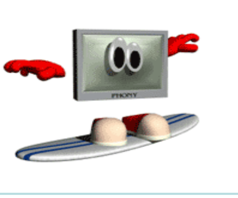
salary S & amount X

1000 300000

W=18000 // العمولة

Allsalary = 19000 // الراتب

Press any key to continue





- اكتب برنامجاً بلغة C# يقوم بإدخال عددين حقيقيين x , y ، وإدخال الرمز ch من نوع char بحيث إذا كان الرمز المدخل هو(+) يقوم بجمع العددين ، وإذا كان (-) يقوم بطرح العددين ، وإذا كان (*) يقوم بطرب العددين ، وإذا كان (/) يقوم بقسمة العددين وطباعة النتيجة .
- ارسم المخطط التدفقي المناسب .



Start

Input x,y,ch

ch = ‘+’ — yes → Z=x + y ; NO →

ch = ‘-’ — yes → Z=x - y ; NO →

ch = ‘*’ — yes → Z=x * y ; NO →

ch = ‘/’ — yes → Z=x / y ; NO → Print خطأ

Print Z

End

```
using System ;

class Arethmatic {
public static void Main(string   [ ] args ) {
 double   x , y , z;
 char   ch ;
 Console.WriteLine("inter yuor x  &  y & ch ");
 x = double.Parse( Console.ReadLine() );
 y = double.Parse( Console.ReadLine() );
ch = char.Parse(Console.Read() ) ;
 Console.WriteLine("--------------------------------");
Console.WriteLine( " x = " +x+"   y="+y) ;
 Console.WriteLine("--------------------------------");
```

```
if( ch == '+')
 Console.WriteLine( x + y );
 //---------------------------
 else if( ch == '-')
 Console.WriteLine( x – y );
 //---------------------------
 else  if( ch == '*')
 Console.WriteLine( x * y );
 //---------------------------
 else if( ch == '/')
 Console.WriteLine( x  / y );
 //---------------------------
 else
 Console.WriteLine( "Sumbol  Erorr Input ");
Console.WriteLine( "--------------------------------");
  } // end method main
} // end class Name
```

RESULT

```
 x  &  y & ch
7      8    *
--------------------------------
 x = 7 y=8
 z= 56
--------------------------------
Press any key to continue
```

أعد كتابة البرنامج باستخدام عبارة طباعة واحدة لطباعة نتيجة العملية الحسابية

```
sing System;
class Arethmatic  {
public static void Main(string[] args) {
   double x, y, z=0;
   char ch;

   Console.WriteLine("inter yuor x  &  y & ch ");
    x = double.Parse(Console.ReadLine());
    y = double.Parse(Console.ReadLine());
    ch = char.Parse(Console.ReadLine());
  if (ch == '+' || ch == '-' || ch == '*' || ch == '/')
  {
      if (ch == '+')
            z = x + y;
        //----------------------------
   else if (ch == '-')
            z = x - y;
//----------------------------
```

```
else if (ch == '*')
            z = x * y;
//-----------------------------
    else if (ch == '/')
            z = x / y;
 //-----------------------------
   Console.WriteLine("--------------------------------");
   Console.WriteLine(" x  " + ch + "   y  =" + z );
   Console.WriteLine("--------------------------------");
}
   else
     {
      Console.WriteLine("Sumbol  Erorr Input ");
      Console.WriteLine("--------------------------------");
        }
    } // end method main
} // end class Name
```

الأسئلة

1. اكتب برنامجاً بلغة C# يقوم بإدخال عدداً حقيقياً من لوحة المفاتيح ؛ ثم يحدد فيما إذا كان هذا العدد موجباً يطبع عبارة (Positive) أم إذا كان سالباً يطبع عبارة (Negative) أما إذا كان مساوياً للصفر يطبع عبارة (Zero) ؛ و يطبع الناتج على الشاشة.
2. اكتب برنامجاً بلغة C# لإيجاد جذور معادلة من الدرجة الثانية :

$$ax^2 + bx + c = 0$$

حيث يطلب من المستخدم ادخال الثوابت a , b , c من لوحة المفاتيح .

3. اكتب برنامجاً بلغة C# لإدخال ثلاثة أعداد حقيقية من لوحة المفاتيح وإيجاد العدد الأكبر (Max number) .
4. اكتب برنامجاً بلغة C# يقوم بإدخال درجة الطالب Degree ويقوم بطباعة كلمة ناجح Passed إذا كانت درجة الطالب $Degree \geq 60$ أو يطبع كلمة راسب Failed .



2- الحلقات
الحلقة for

- تستخدم الحلقة for لتكرار مجموعة من التعليمات عدداً محدداً من المرات , **وتحوي على ثلاثة أقسام هي : التهيئة والتعبير والتحديث .**
- **التهيئة :** هو تعبير يقوم بتهيئة متحول الحلقة initialization.
- **التعبير** termination : فيمثل شرط التوقف ويتم اختياره في بداية كل تكرار ويحدث التكرار فقط في حال كانت قيمة التعبير هي true
- **التحديث increment :** يمثل مقدار الزيادة على متحول الحلقة.
- **الصيغة العامة :**

```
for(init exp ; test exp ; inc exp)
        <statement>;
```

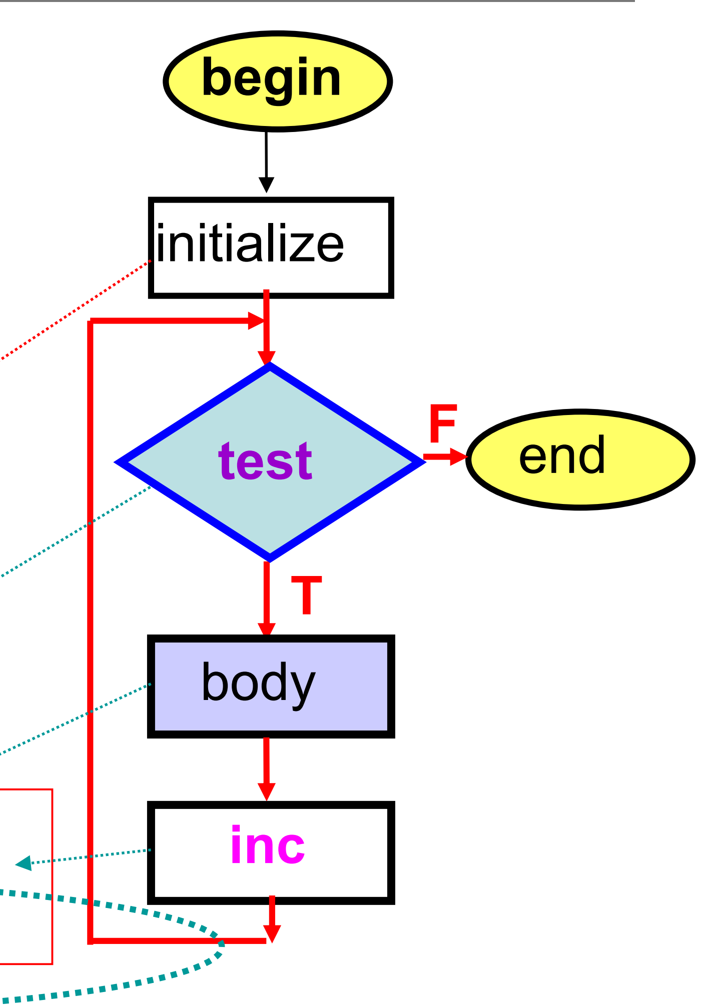
or compound statement (block) { }



Steps مراحل تنفيذ الحلقة

- Initialize
- **Test**
  - ✓False, don’t execute
  - ✓True, execute
- Increment counter

For(init exp ; test exp ; inc exp**)**
<statement>;



for examples

```
for (i=0; i<10; i++)
  Console.WriteLine ("i is " + i);

for (int j=10; j>=0; j=j-2 ) {
  Console.WriteLine ("j is " + j);
   if (j<6)
       break;
}
```





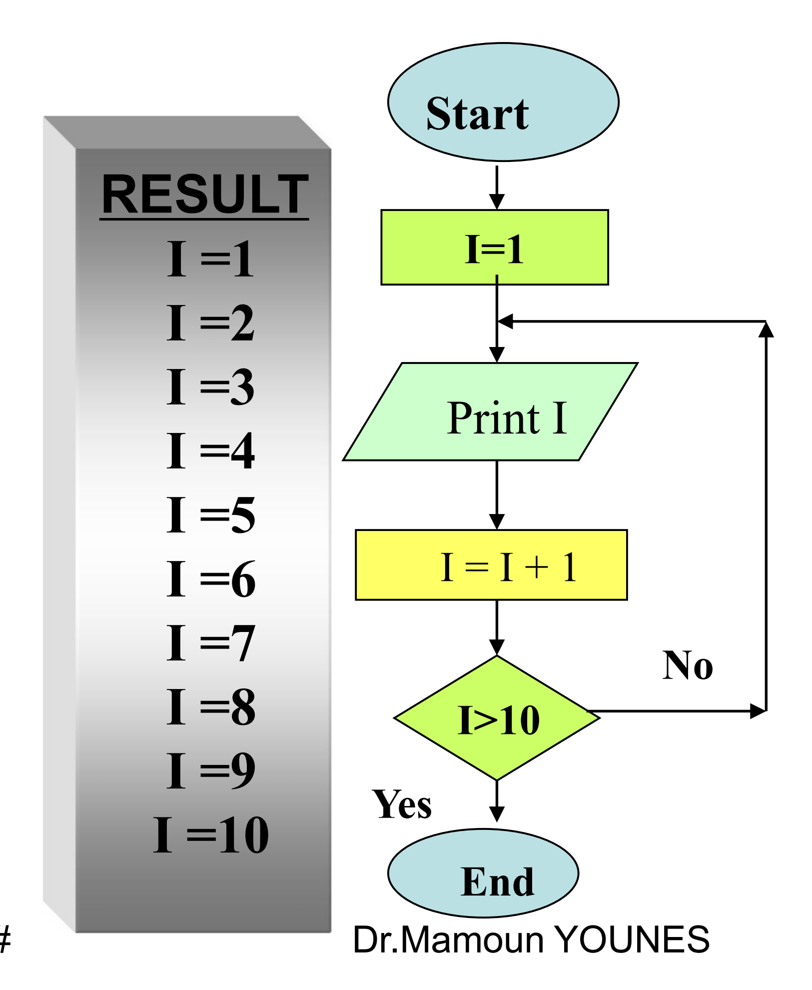
Example:

اكتــــب خوارزميــــة لطبــــع الاعــــداد الصحيحة من 1 الى 10 .

```
using System ;

class Increment {
public static void Main(string
    [ ] args ) {

for ( int i=1;i<=10 ; i++ )
  {
 Console.WriteLine(" i = {0}" ,i ) ;
  }//End for
 } // end method main
} // end class Name
```





مثال(24)

Example:

اكتب خوارزمية لطبع الاعداد الصحيحة الزوجية من 1 الى 10

```
using System ;

class Increment {
public static void Main(string   [ ] args ) {

for ( int i=2;i<=10 ; i=i+2 )
  {
 Console.WriteLine("i = {0}" ,i " , i ) ;
  }//End for
 } // end method main
} // end class Name
```

RESULT

I =2
I =4
I =6
I =8
I =10

Start → I=2 → Print I → I = I + 2 → I>=10 (No → back to Print I; Yes → End)





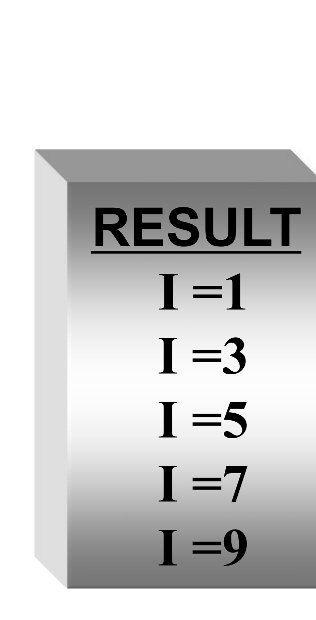
Example:

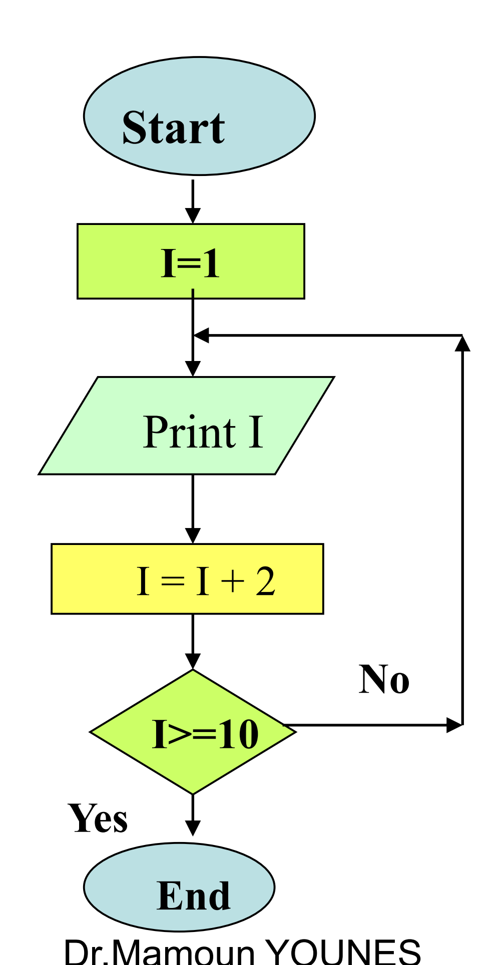
اكتب خوارزمية لطبع الاعداد الصحيحة الفردية من 1 الى 10 .

```
using System ;

class Increment {
public static void Main(string    [ ] args ) {

for ( int i=1;i<=10 ; i=i+2 )
  {
 Console.WriteLine("i = {0}" , i ) ;
  }//End for
 } // end method main
} // end class Name
```









المثال التالي يوضح التعليمة **for**.

حيث يتم حساب مجموع الأعداد من 0إلى 9 .

```
// Add the numbers from 0 to 9 and print the result
using System ;
class Increment {
public static void Main(string [ ] args ) {
int sum = 0;
   for ( int i = 0; i <= 9; i++)
          sum = sum + i; //End for
 Console.WriteLine(" sum =" +sum ) ;

 } // end method main
} // end class Name
```

RESULT

Sum = 45.

Press any key to continue

.



أحسب مجموع عشر أعداد حقيقية مختلفة القيم تدخل من قبل المستخدم . ثم أحسب المتوسط

```
// Add 10 numbers and print the average
using System ;

class average {
public static void Main(string [ ] args ) {
double N ,avr , sum = 0;
    for (int  i = 0; i <= 9; i++){
     N = double.Parse( Console.ReadLine() );
    sum = sum +N;
    } //End for
    avr = sum / 10 ;
Console.WriteLine(" sum =" +sum+"  are= "+avr ) ;
 } // end method main
  } // end class Name
```

Start

I=1

Input N

sum= sum +N

I = I + 1

I<=10

Yes

Print sum , sum/10

No

End

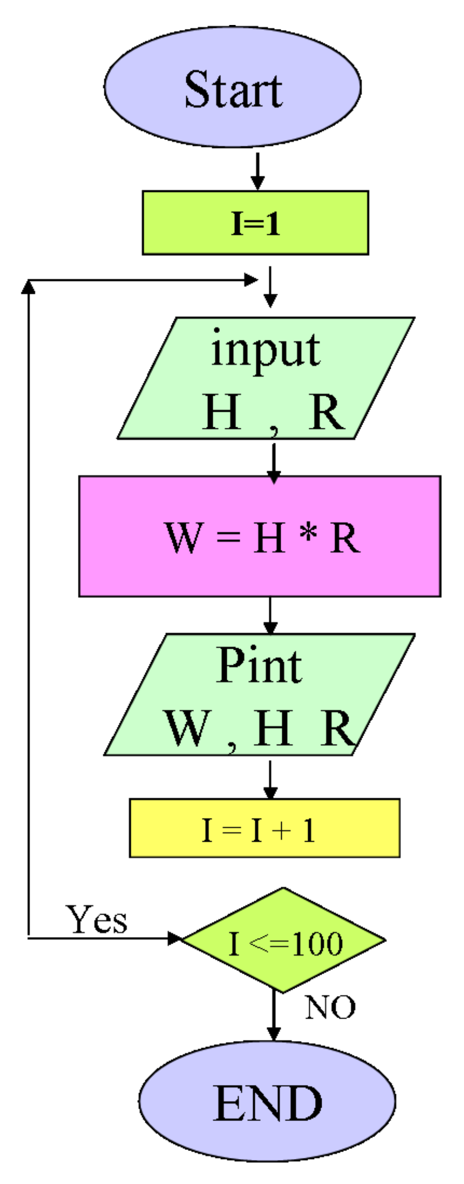


- **يعمل 100عامل في معمل نسيج بشكل ساعي , لنفرض أن H هي عدد ساعات العمل وأن R هي أجرة الساعة الواحدة وأن W هو الأجر اليومي . والمطلوب :**
- اكتب برنامجاً بلغة #C يقوم بما يلي :
  - قراءة وطباعة عدد ساعات العمل وأجرة الساعة .
  - حساب وطباعة الأجر اليومي لكل عامل .
- ارسم المخطط التدفقي المناسب

```
using System ;

class Worker {
public static void Main(string [ ] args) {
double  H , R ,W;

for (int  i = 1; i <= 5; i++){
    Console.WriteLine(" Enter R & H " ) ;
    R = double.Parse( Console.ReadLine() );
    H = double.Parse( Console.ReadLine() );
        W = R * H ;
    Console.WriteLine(" W = {0}  " , W ) ;
    }//End for
  } // end method main
} // end class Name
```





الحلقة while

- تُستخدم التعليمة while لتكرار مجموعة من التعليمات طالما أن الشرط محقق .

Loop expression

Body of the loop

while (<*condition*>)
- <statement>;
- or compound statement (block) **{ }**



Test the expression

0 *or* False

Exit the *while*

1 *or* True

statements to execute

loop

مثال(28)

المثال التالي يوضح التعليمة **while**.
حيث يتم حساب مجموع الأعداد من 0إلى 9 .

```
// Add the numbers from 0 to 9 and print the result while
    loop example
using System ;

class SumNumber {
public static void Main(string [ ] args ) {
int i=0, sum = 0;
    while(i <= 9) {
        sum = sum + i; // sum += i;  accumulator
         i++;  } // i = i + 1;  counter
Console.WriteLine("Sum = " + sum + ".\n");
  } // end method main
} // end class Name
```

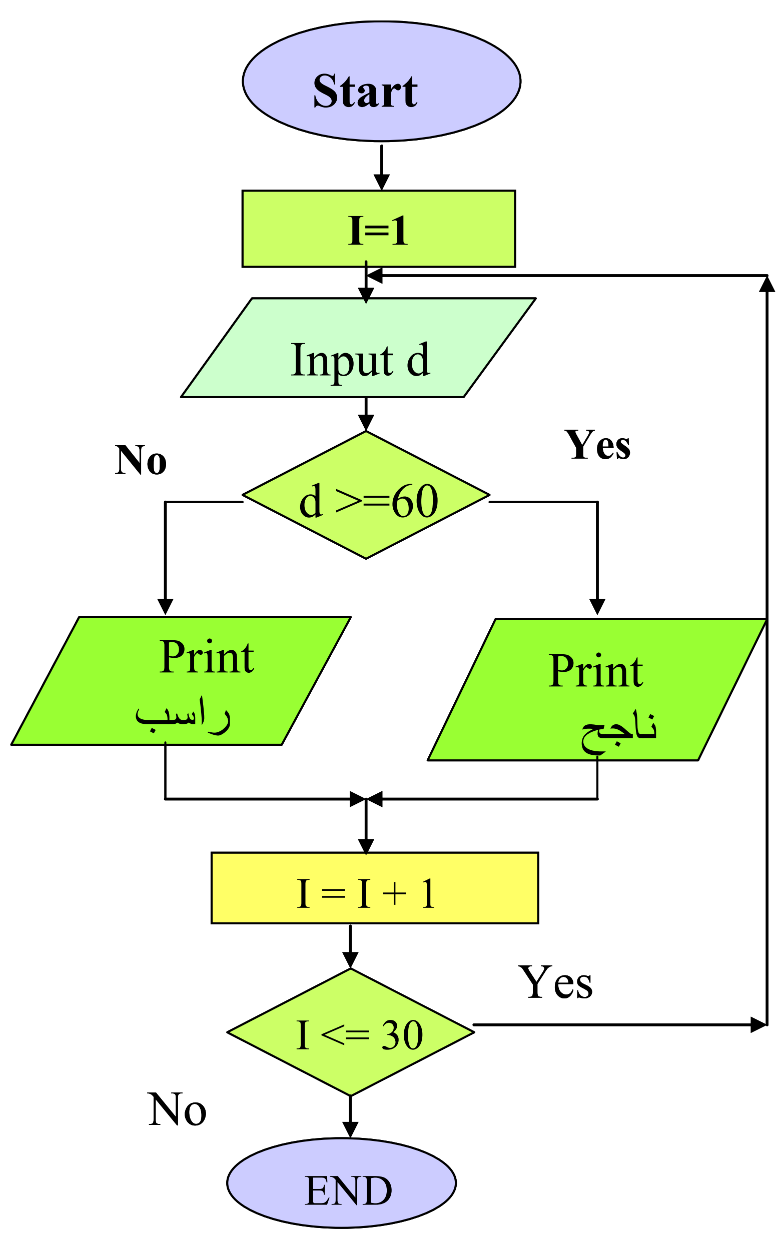
RESULT

Sum = 45.

Press any key to continue



- لـديـنـا 30 طالبـاً يدرسـون مقـرر البرمجـة والمطلـوب : اكتب برنامجاً بلغة #C يقوم بما يلي :
  - إدخـال درجـة الطالـب d وطباعـة كلمـة نـاجح" Passed " إذا كانت درجة الطالب 60 ≤ d ويطبع كلمة راسب " Failed " إذا كانت غير ذلك .
- ارسم المخطط التدفقي اللازم .
- استخدم الحلقة **while**

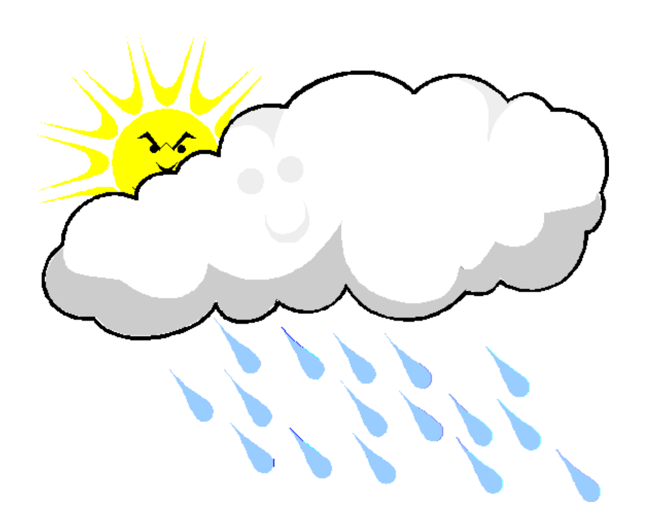

Start
I=1
Input d
No
d >=60
Yes
Print
راسب
Print
ناجح
I = I + 1
Yes
I <= 30
No
END

```
 using System ;
class Student {
public static void Main(string [ ] args ) {
double d;
int i=1;  // القيمة الابتدائية
 while(i <= 30)  // الشرط
  {
 Console.WriteLine(" Enter degree ");
 d = double.Parse( Console.ReadLine() );
  if ( d >= 60 )
 Console.WriteLine(" Passed = {0}“ ,d);
  else
 Console.WriteLine(" Failed = {0}“ ,d);
        i++;  } // i = i + 1;  counter
  } // end method main
} // end class Name
```

مثال(29-b)

- يقوم البرنامج التالي بإدخال وحساب مجموع عدد غير محدد من الأرقام الصحيحة .
  - عند ادخال القيمة 0 (صفر) يشير إلى نهاية البرنامج .

```
using System ;
class Student {
public static void Main(string [ ] args ) {
    int data;
    int sum = 0;
    Console.WriteLine(" Enter the first No. ");
    data = int.Parse(Console.ReadLine());
//--------  start while ---------------
 while(data != 0) {  // الشرط
      sum = sum + data;
 Console.WriteLine(" Enter integer No. ");
 data = int.Parse(Console.ReadLine());
 }// end while ----------------------
Console.WriteLine(" Sum = {0}" ,sum);
  } // end method main
} // end class Name
```



RESULT

Enter the first No.
45
 Enter integer No.
66
 Enter integer No.
7
 Enter integer No.
88
 Enter integer No.
0
 Sum = 206
Press any key to continue . . .

مثال(29-c)

- يقوم البرنامج التالي بإدخال عدد صحيح n موجب وحساب العاملي لهذا العدد n وطباعة النتيجة .
  - عند إدخال عدد سالب يقوم بطباعة العبارة

Error: n is a negative number

  - عندما يكون العدد صفر أو واحد يطبع النتيجة Fact = 1
  - أما إذا كان العدد أكبر من الواحد يقوم بحساب وطباعة العاملي

```
using System;
class Factorail  {
static void Main(string[] args) {
  double n, Fact = 1;
Console.WriteLine("Enter an positive integer to find its factorial: ");
  n = double.Parse(Console.ReadLine());
 if (n < 0)
Console.WriteLine( "Error: " + n + " is a negative number" );

else if ((n == 0) || (n == 1))
   Console.WriteLine( n + "! = " + Fact );
```

```
else
  {
    int i = 1;
    while (i <= n) {
         Fact = Fact * i;
           i++;
}  //end while
Console.WriteLine( n + "! = " + Fact );
            }

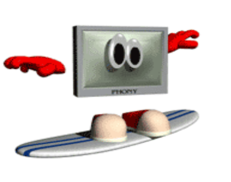
        }//end main
    }//end class
```



مقارنة بين الحلقة for والحلقة while
Compare while and for

```
int i = 0 , sum = 0;
  while (i < 9){
        sum = sum + i;
        i++;
{
```

```
sum = 0;
 for( i = 0; i < 9; i++)
       sum = sum + i;
```



do - while التعليمة

- تعتبـر هـذه الحلقـة كمرادفـة للحلقـة while إلا أن الشـرط سيختبر في نهاية الحلقة بدلاً من بدايتها .
- يتم تنفيذ الحلقة مرة واحدة علـى الأقل وتستمر عمليـة التنفيذ طالما أن الشرط محقق true .
- **الصيغة العامة :**

```
do
    statement;
while ( condition )
```



```
do
   statement;
while ( condition )
```

```
i=0; x=0;
do {
  x += i;
  i++;
} while ( i<10 );
```



المثال التالي يوضح التعليمة **do - while**.
حيث يتم حساب مجموع الأعداد من 0إلى 9 .

```
// Add the numbers from 0 to 9 and print the result do while loop example
using System ;

class DoWhile {
public static void Main(string  [ ] args ) {
int i=0, sum = 0;
    do {
       sum = sum + i;  // sum += i; accumulator
        i++; } // i = i + 1;  counter
          while(i <= 9) ;
Console.WriteLine("Sum = " + sum + ".\n");
  } // end method main
} // end class Name
```

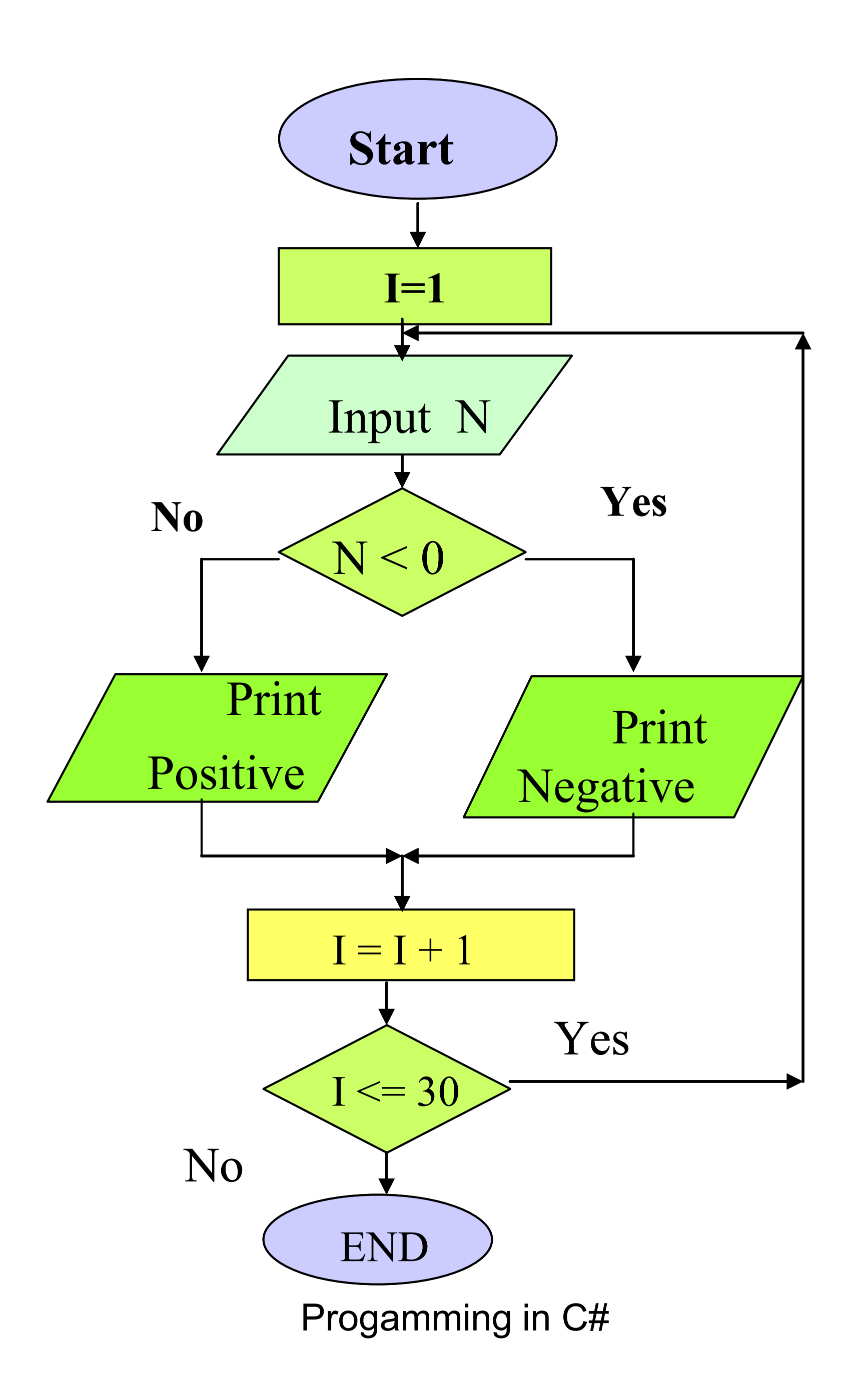
RESULT

Sum = 45.

Press any key to continue



- **اكتب برنامجاً بلغة C# يقوم بإدخال 30 عدداً حقيقياً** وطباعة كلمة " Negative " إذا كان العدد أصغر من الصفر , ويطبع كلمة موجب إذا كان العدد أكبر أو يساوي الصفر .
- ارسم المخطط التدفقي المناسب .
- استخدم الحلقة **do - while**


Start
I=1
Input N
No
Yes
N < 0
Print Positive
Print Negative
I = I + 1
Yes
I <= 30
No
END

```
using System ;

class Number {
public static void Main(string  [ ] args ) {
double number;
int i=1;  // القيمة الابتدائية
 do {
 Console.WriteLine(" Enter degree ");
 number = double.Parse( Console.ReadLine() );
  if (number >= 60 )
 Console.WriteLine(" Positive = {0}“, number );
  else
```

```
Console.WriteLine(" Negative = {0}“, number );
        i++;  } // i = i + 1;  counter
while(i <= 30) ; //  الشرط
} // end method main
  } // end class Name
```

مثال(32)

- في شركة 30 مندوب مبيعات يتقاضى كل مندوب راتباً شهرياً أساسيا مقداره (000 10) ليرة , ويتقاضى عمولة قدرها 4% إذا كانت المبيعات الشهرية لا تتجاوز (200 000) ليرة , ويتقاضى عمولة قدرها 6% إذا كانت المبيعات الشهرية لا تتجاوز (400 000) ليرة, ويتقاضى عمولة قدرها 7% إذا كانت المبيعات الشهرية أكثر من ذلك . والمطلوب :
- اكتب برنامجاً بلغة C# يقوم بما يلي :
- ادخال الراتب الشهري وقيمة المبيعات الشهرية وطباعتها . salary ,W ,Allsalary, amont
- حساب وطباعة الدخل الشهري الكامل للموظف .
- ارسم المخطط التدفقي المناسب .
- ملاحظة : افترض salary الراتب الشهري الأساسي , amont المبيعات الشهرية , W العمولة , Allsalary الراتب الشهري الكامل .

```
using System ;

class Ssalary {
public static void Main(string  [ ] args ) {
  int  salary   , amont ;
  double  W ,Allsalary ;
for ( int i=1 ; i<= 3 ; i++ )
  {
 Console.WriteLine("inter yuor salary & amont");
salary = double.Parse( Console.ReadLine() );
amont = double.Parse( Console.ReadLine() );
Console.WriteLine("------------------------------");
if( amont <= 200000 )
      W=amont*4/100;
```

```
else if( amont <= 400000 )
      W=amont*6/100;
  else
      W=amont*7/100;
Console.WriteLine("W= {0}"  , W);
Console.WriteLine("------------------------------");
Allsalary = salary + W ;
 Console.WriteLine("Allsalary = {0} " , Allsalary);
 Console.WriteLine("------------------------------");
}//End for
  } // end method main
} // end class Name
```



تعليمات التفريع
Branching Statement

1- تعليمة break :

- تستخدم هذه التعليمة لإنهاء الحلقات التكرارية .

مثال(33)

```
using System ;
public class break1 {
  public static void Main(string [ ] args ) {
   int x;
```

```
for ( x = 1;  x<= 10; x++ )
   {
        if ( x == 4 )
           break ;
        Console.WriteLine("x= {0}" , x);
   }
   Console.WriteLine("\nBroke out of loop at x == " + x+ "\n");

  } // end method main
} // end class Name
```

RESULT

x= 1

x= 2

x= 3

Broke out of loop at x == 4

Press any key to continue

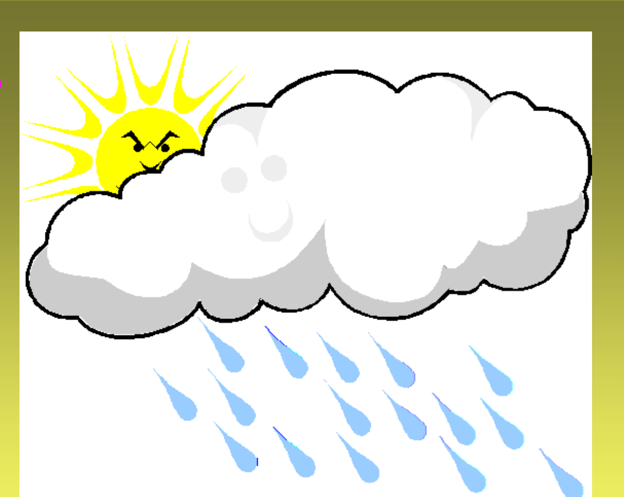



```
// Demonstrate break statement.
using System ;
class breakTest {
public static void Main(string  [ ] args ) {
for (int i=0;i<10;i++) {
 Console.WriteLine(" i  is " + i);
   if (i==3) break; }
outer: for (int j=0;j<5;j++) {
   for (int k=0;k<5;k++) {
     if (k==3)  goto  outer;
 Console.WriteLine("j,k: " + j + "," + k);
   }
}
  } // end method main
} // end class Name
```

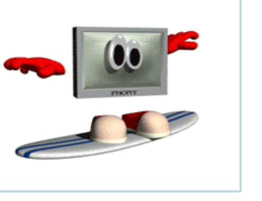


RESULT

i is 0
i is 1
i is 2
i is 3
j,k: 0,0
j,k: 0,1
j,k: 0,2
j,k: 1,0
j,k: 1,1
j,k: 1,2
j,k: 2,0
j,k: 2,1
j,k: 2,2
j,k: 3,0
j,k: 3,1
j,k: 3,2
j,k: 4,0
j,k: 4,1
j,k: 4,2



Press any key to continue



2- تعليمة continue

- تستخدم لإهمال تنفيذ ما تبقى من تعليمات الحلقة والانتقال إلى تكرار جديد .




```
using System ;
public class continue1 {
public static void Main(string [ ] args ) {
   int x;
```

```
for ( x = 1;  x<= 10; x++ )
   {
    if ( x == 5 )
      continue ;
    Console.WriteLine("x= {0} " , x);
   }
    Console.WriteLine("\n Used continue
                          to skip printing the value 5 " );

  } // end method main
} // end class Name
```

RESULT

x= 1

x= 2

x= 3

x= 4

x= 6

x= 7

x= 8

x= 9

x= 10

Used continue to skip printing the value 5

Press any key to continue

.




```
// Demonstrate break statement.
using System ;
class ContinueTest {
public static void Main(string [ ] args ) {
for (int i=0;i<10;i++) {
if (i==3) continue;
 Console.WriteLine("i is " + i); }
for (int j=0;j<5;j++) {
   for (int k=0;k<5;k++) {
     if (k==3) continue;
 Console.WriteLine("j,k: " + j + "," + k);
   }
}
  } // end method main
} // end class Name
```



RESULT

```
i is 0
i is 1
i is 2
……
i is 4
i is 5
i is 6
i is 7
i is 8
i is 9
```

Press any key to continue

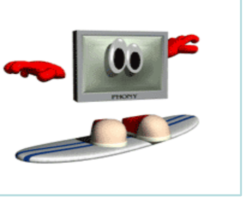
```
j,k: 0,0
j,k: 0,1
j,k: 0,2
j,k: 1,0
j,k: 1,1
j,k: 1,2
j,k: 2,0
j,k: 2,1
j,k: 2,2
j,k: 3,0
j,k: 3,1
j,k: 3,2
j,k: 4,0
j,k: 4,1
j,k: 4,2
```

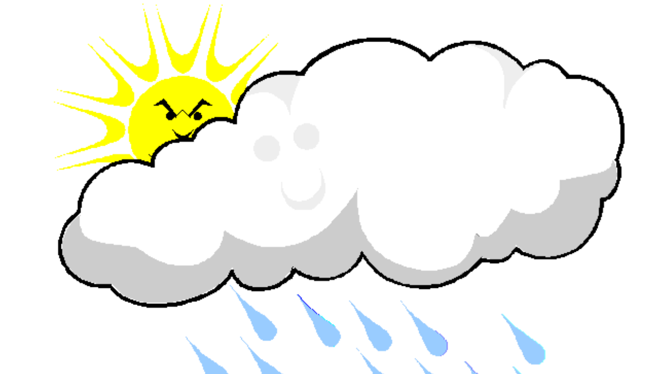


3- تعليمة return

- تُستخدم للخروج من التابع الحالي , ويُعاد تحكم التنفيذ إلى التعليمة التي تلي تعليمة استدعاء التابع مباشرة .
- تملك التعليمة return شكلين :
  - الأول : يُعيد قيمة (كما هي في التوابع التي تعيد قيمة) .
  - الثاني : لا يعيد قيمة .
- لإعادة قيمة يكفي أن نضع القيمة المراد إعادتها بعد العبارة return مباشرة . مثال :

```
return ++count ;
```



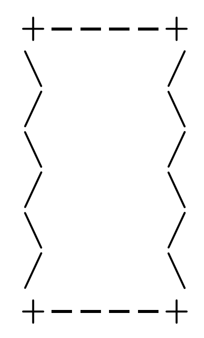


- يجب على نمط القيمة المعادة أن يطابق نمط التابع .
- عند التصريح عن التابع على أنه void , لا داعي لإعادة أي قيمة ويكفي استخدام العبارة return فقط للخروج من التابع كما في الشكل التالي :

return ;

الأسئلة

1. Write a program that prints the following shape .



2. Write a program that prints the following shape .

```
+/\/\/\/\/\+
|          |
+/\/\/\/\/\+

+/\/\/\/\/\+
|          |
|          |
|          |
|          |
|          |
+/\/\/\/\/\+
```

3. Write a program that prints the following shape

```
*
**
***
****
*****
******
```

4. Write a program that prints the following shape

a

```
1
22
333
4444
55555
666666
```

b

```
    1
   2
  3
 4
5
```

c

```
    1
   22
  333
 4444
55555
```

5.Write a program that prints the following shape

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
```

6.

مسائل عامة

1. اكتب برنامجاً بلغة C# يقوم بما يلي :
   - إدخال عددين صحيحين a و b .
   - إيجاد وطباعة المضاعف المشترك الأصغر للعددين المُدخلين.
2. اكتب برنامجاً بلغة C++ يقوم بما يلي:
   - إدخال عددين صحيحين a و b.
   - إدخال قيمة صحيحة للمتحول m بحيث إذا كانت m=1 يقوم البرنامج بإيجاد العدد الأكبر من بين العددين المُدخلين , وإذا كانت m=2 يقوم البرنامج بحساب المتوسط الحسابي للعددين المُدخلين , وإذا كانت m=3 يقوم البرنامج بإيجاد المضاعف المشترك الأصغر للعددين المُدخلين .
   - استخدم الحلقة switch.
3. اكتب برنامجاً بلغة c++ يقوم بإيجاد وطباعة درجة الحرارة العظمى خلال شهر تشرين الأول (31 يوم)، باستخدام الحلقة while .

4. اكتب برنامجاً بلغة C# يقوم بإدخال 50 عدداً صحيحاً وحساب وطباعة المتوسط الحسابي للأعداد الصحيحة الفردية .

5. البرنامج التالي يستخدم حلقة التكرار while والتي من خلالها يُطلب من المستخدم الاستمرار بإدخال محرف ما ويتم التوقف عن التكرار عندما يقوم المستخدم بإدخال المحرف Y أو المحرف N .

الحل

1.

```
Console.WriteLine("+----+");
for (int i = 1; i <= 3; i++) {
 Console.WriteLine("\\    /");
 Console.WriteLine("/    \\");
}
Console.WriteLine("+----+");
```

================================

3.

```
   for (int i = 1; i <= 6; i++) {
       for (int j = 1; j <= i; j++) {
    Console.Write("*");
       }
    Console.WriteLine();
   }
```

4-a :

```
for (int i = 1; i <= 6; i++) {
    for (int j = 1; j <= i; j++) {
 Console.Write(i);
    }
 Console.WriteLine();
}
```

4-b :

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= (5 - i); j++) {
 Console.Write(" ");
    }
 Console.WriteLine(i);
}
```

4-c :

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= (5 - i); j++) {
        Console.Write(" ");
    }
    for (int k = 1; k <= i; k++) {
        Console.Write(i);
    }
    Console.WriteLine();
}
```

==============================

5.

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 10; j++) {
        Console.Write((i * j) + " ");
    }
    Console.WriteLine();  // to end the line
}
```

the end

The End

The End

The End

The End

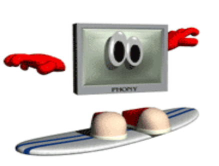
The

End

The End

The End


The End
The End



البرمجة بلغة



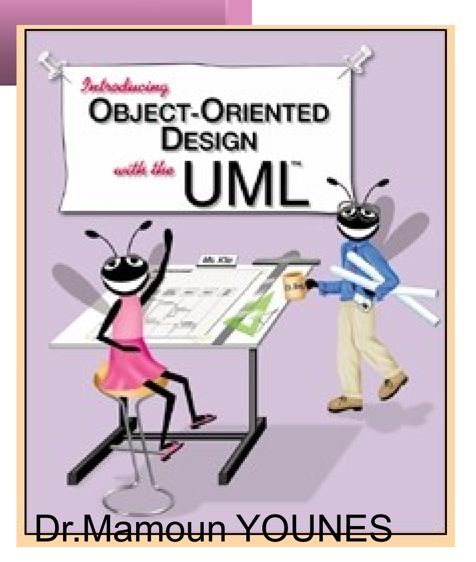


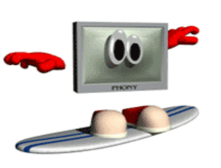
Programming

Dr. M.Younes









التوابع

Functions

Chapter 3

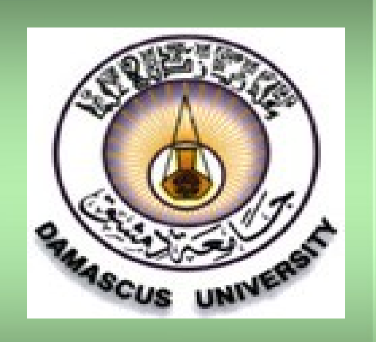


محتويات الفصل الثاني

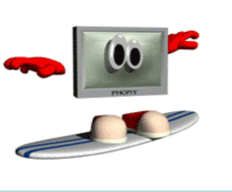
❑ التوابع Methods

- ❖ فوائد استخدام التوابع
- ❖ نموذج التابع
- ❖ استدعاء التوابع
- ❖ مكتبة الدوال الرياضية
- ❖ التوابع بدون وسطاء
- ❖ التمرير بالقيمة والعنوان (المرجع)
- ❖ التحميل الزائد للتوابع
- ❖ تابع العودية والعشوائي

التوابع
Functions



- بنية التابع
- استدعاء التابع



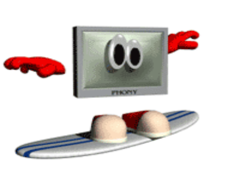
التوابع Methods

- توجد في لغة #C مكتبة ضخمة من التوابع تقوم بتنفيذ العمليات الرياضية، والتعامل مع السلاسل والمحارف، والإدخال والإخراج، واكتشاف الأخطاء والعديد من العمليات الأخرى المفيدة مما يسهل مهمة المبرمج الذي يجد في هذه التوابع معيناً كبيراً له في عملية البرمجة.
- **يمكن للمبرمج كتابة توابع تقوم بأداء عمليات يحتاج لها المبرمج في برامجه .**



فوائد استخدام التوابع في البرمجة

1. تساعد التوابع المخزنة في ذاكرة الحاسوب على اختصار البرنامج إذ يكتفي باستدعائها باسمها فقط لتقوم بالعمل المطلوب .
2. تساعد البرامج المخزنة في ذاكرة الحاسوب أو التي يكتبها المستخدم على تلافى عمليات التكرار في خطوات البرنامج التي تتطلب عملاً مشابهاً لعمل تلك التوابع .
3. تساعد التوابع الجاهزة في تسهيل عملية البرمجة.
4. يوفر استعمال التوابع من المساحات المستخدمة في الذاكرة.
5. كتابة برنامج في لغة الـ C# بشكل تابع واضحة المعالم يجعل البرنامج واضحاً لكل من المبرمج والقارئ على حد سواء.



- **التوابع تمكن المبرمج من تقسيم البرنامج إلى وحدات modules**، كل تابع في البرنامج يمثل وحدة قائمة بذاتها، ولذا نجد أن المتغيرات المعرفة في التابع تكون متغيرات محلية (Local) ونعنى بذلك أن المتغيرات تكون معروفة فقط داخل التابع.
- **أغلب التوابع تمتلك لائحة من الوسائط (Parameters)** والتي هي أيضاً متغيرات محلية .



نموذج التابع
Method Prototype

- عندما يولد المترجم تعليمات لاستدعاء تابع ما فإنه يحتاج إلى معرفة اسم التابع وعدد وسطائه وأنواعها ونوع قيمة الإعادة .
- **لذا علينا كتابة نموذج أو (تصريح) للتابع ضمن الصف الذي يحتوي على التابع Main (قبل التابع Main أو بعده) .**
- **عندما نصرح عن تابع ما** فإننا نقوم فعلياً **بتعريف مهمة هذا التابع** و يبلغ المترجم عن اسم التابع وعدد وسطائه وأنواعها ونوع القيمة المعادة بواسطة التابع.
- فمثلا في تصريح التابع التالي:

```
public static int   sum( int x )  {  جسم التابع  }
```

النوع **int** بين القوسين يخبر المترجم بأن الوسيط الذي سيتم تمريره إلى التابع سيكون من النوع int و **int** التي تسبق اسم التابع تشير إلى نوع القيمة المعادة بواسطة التابع ، ويمكن أن تكون من نمط double أو char كما يلي :

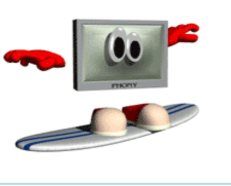
```
public static double  sum1(double  x ) { . . . }
public static char  sum2( char  x ){ . . . }
```

- أو لا يعيد قيمة وهو من نمط **void**

```
public static void sum( int  x , double  y ) {. . .}
```

- أو بدون وسطاء :

```
public static void sum( ) {. . .}
```

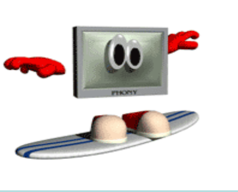


- يأخذ تعريف التابع في #C الشكل العام التالي:

```
public static type  function-name ( parameter list)
   {
        declarations and statements
   }
```

حيث:

- **type** : نوع القيمة المعادة بواسطة التابع والذي يمكن أن يكون أي نوع من أنواع معطيات #C ، وإذا كان التابع لا يرجع أي قيمة يكون نوع إعادته void.
- **function-name:** اسم التابع والذي يتبع في تسميته قواعد تسمية المعرفات (identifiers).



- **parameter list:** هي لائحة الوسطاء الممرة إلى التابع وهي يمكن أن تكون خالية (void) أو تحتوى على وسيط واحد أو عدة وسطاء تفصل بينها فاصلة ويجب ذكر نوع كل وسيط على حدة.
- **declarations and statements:** تمثل جسم التابع والذي يطلق عليه في بعض الأحيان block .
- **ويمكن أن يحتوى الـ block على تصريح المتغيرات ولكن تحت أي ظرف لا يمكن أن يتم تعريف تابع داخل جسم تابع آخر.**
- السطر الأول في تعريف التابع يدعى المصرح declarator والذي يحدد اسم التابع ونوع المعطيات الذي يعيدها التابع وأسماء وأنواع وسطائه .



استدعاء التوابع
Method Call

- يؤدى استدعاء التابع إلى انتقال التنفيذ إلى بداية التابع.
- يمكن تمرير بعض الوسطاء إلى التابع عند استدعائه وبعد تنفيذ التابع يعود التنفيذ للعبارة التي تلي استدعاء التابع.
- بإمكان التابع أن يعيد قيم إلى العبارة التي استدعاه ، ويجب أن يسبق اسم التابع في معرفه .
- وإذا كان التابع لا يعيد شيئاً يجب استعمال الكلمة المفتاحية void كنوع إعادة له للإشارة إلى ذلك .

- **هنالك ثلاث طرق يمكن بها إرجاع التحكم إلى النقطة التي تم فيها استدعاء التابع :**

1. **إذا كان التابع لا يرجع قيمة** يرجع التحكم تلقائياً عند الوصول إلى نهاية التابع .
2. **باستخدام العبارة** return ;
3. **إذا كان التابع يرجع قيمة** فالعبارة return expression ; تقوم بإرجاع قيمة التعبير expression إلى النقطة التي استدعته .

- لنأخذ برنامجاً يستخدم تابع يدعى square لحساب مربعات الأعداد من 1 إلى 10.

مثال(37)

```
using System ;

class SguareNumber {
public static void Main(string [ ] args ) {
    for (int  x = 1; x <= 10; x++){
            Console.Write (" {0}   " ,sguare(x) ) ;
    }
     Console.WriteLine( ) ;
//End for
} // end method main
//now function definition
  public static int sguare( int y)
{
    return y*y;
}
} // end class Name
```

RESULT

1 4 9 16 25 36 49 64 81 100

نعيد كتابة البرنامج السابق باستخدام تابع لا يعيد قيمة




```
using System ;
class SguareNumber {
public static void Main(string [ ] args ) {
    for (int  x = 1; x <= 10; x++){
            sguare(x) ;
    }
     Console.WriteLine( ) ;
//End for
} // end method main
//now function definition
  public static void sguare( int y)
{
     Console.Write(“ {0}   “ ,y*y ) ;
}
} // end class Name
```

RESULT

1 4 9 16 25 36 49 64 81 100

Press any key to continue ……..





- البرنامج التالي يستخدم تابع يدعى maximum والذي يرجع العدد الأكبر بين ثلاثة أعداد صحيحة.
- يتم تمرير الأعداد كوسائط للتابع الذي يحدد العدد الأكبر بينهم ويرجعه للتابع main باستخدام العبارة return ويتم تعيين القيمة التي تمت إعادتها إلى المتغير largest الذي تتم طباعته.

```
using System ;

class Maximum {
public static void Main(string [ ] args ) {
     double  a, b, c;
      Console.WriteLine("Enter three integers: ") ;
     a = double.Parse( Console.ReadLine() );
    b = double.Parse( Console.ReadLine() );
    c = double.Parse( Console.ReadLine() );
     Console.WriteLine(" maximum ={0} " , maximum (a, b, c) ) ;

}// // end method main
```

```
public static double  maximum (double x, double y, double z)
{
   double  max = x;
    if (y > max)
    max = y;
    if (z > max)
    max = z;
//Continued
   return   max;
}//end max
} // end class Name
```

RESULT

Enter three integers: 22 85 17

Maximum is: 85

Press any key to continue

نعيد كتابة البرنامج السابق باستخدام تابع لا يعيد قيمة

مثال(40)

```
using System ;

class Maximum {
public static void Main(string [ ] args ) {
      double  a, b, c;
       Console.WriteLine("Enter three integers: ") ;
     a = double.Parse( Console.ReadLine() );
     b = double.Parse( Console.ReadLine() );
     c = double.Parse( Console.ReadLine() );
      maximum (a, b, c)    ;

}// // end method main
```

RESULT

Enter three integers: 22 85 17
Maximum is: 85

```
public static void  maximum (double x, double y, double z)
{
   double  max = x;
    if (y > x)
    max = y;
    if (z > max)
    max = z;
//Continued
 Console.WriteLine(" maximum= {0}  " , max) ;
}//end max
} // end class Name
```

ملاحظة

- من غير الضروري أن تكون أسماء الوسطاء في التصريح عند استدعاء التابع هي نفسها المستعملة في تعريف التابع .
- في الواقع، المترجم يتجاهلها لكنها تكون مفيدة أحياناً للذين يقرؤون البرنامج . فمثلاً لنفترض أن الوسيطين x و y تمثلان إحداثيات نقطة على الشاشة.

[التصريح عند استدعاء التابع] draw_dot (x, y) ;

- هذا التصريح كافي للمعرف لكن المبرمج قد لا يعرف أيهما الإحداثي x وأيهما الإحداثي yعند تعريف التابع إذا استخدم وسطاء تختلف عن وسطاء استدعاء التابع . لذا سيكون مفيداً لو كتبنا :

```
public static void draw_dot (int x, int  y) {   }
```

مثال(40-a)

البرنامج التالي يقوم بحساب مجموع الأعداد الصحيحة من 1 حتى 10 ، ومن 20 حتى 30 ومن 35 حتى 45 باستدعاء التابع sum أكثر من مرة .

```
using System;

class Program {

static void Main(string[] args) {
    Console.WriteLine("sum from 1 to 10 = "+ sum(1,10)) ;
    Console.WriteLine("sum from 20 to 30 = "+sum(20,30) ) ;
    Console.WriteLine("sum from 35 to 45 = " +sum(35,45)) ;

        } // end method main
```

```
public static int sum ( int i1 , int i2) {
            int sum1 = 0;
            for (int i=i1; i<= i2 ;i++)
            sum1 = sum1 + i ;
            return sum1 ;
      } // end class
```

RESULT

```
sum from 1 to 10 = 55
sum from 20 to 30 = 275
sum from 35 to 45 = 440
Press any key to continue
```

تمارين عامة(1)

1. أنشئ تابعاً بلغة #C اسمه ( )sumsqaure يقوم بحساب مجموع مربعات الأعداد من 1 إلى 10.ثم اكتب برنامجاً يقوم باستدعاء هذا التابع وطباعة النتيجة .
2. أنشئ تابعاً بلغة #C اسمه ( )sumpow يقوم بحساب مجموع السلسلة $\sum_{i=1}^{i=10} i^{r}$ =z.ثم اكتب برنامجاً يقوم باستدعاء هذا التابع وطباعة النتيجة .
3. أنشئ تابعاً بلغة #C اسمه ( )Cube يقوم بحساب حجم مكعب طول ضلعه x .ثم اكتب برنامجاً يقوم باستدعاء هذا التابع وطباعة النتيجة .

مكتبة الدوال الرياضية (ضمن الصف Math)

Math Library Methods

- تحتوى مكتبة التوابع الرياضية على العديد من التوابع التي تستخدم في تنفيذ العمليات الرياضية الحسابية. فمثلاً المبرمج الذي يرغب في حساب وطباعة الجذر التربيعي للعدد 900 قد يكتب عبارة كالتالية :

```
number = Math.Sqrt ( 900 );
Console.WriteLine( “ {0} “, number ) ;
```

عند تنفيذ هذه العبارة يتم استدعاء التابع المكتبي Sqrt من الصف **Math** لحساب الجذر التربيعي للعدد بين القوسين (900).

- ويسمى العدد بين القوسين وسيط التابع argument وعليه فالعبارة السابقة تقوم بطباعة العدد 30 .
- ويأخذ التابع Sqrt وسيط من النوع double وتكون النتيجة قيمة من نفس النوع وينطبق هذا على جميع التوابع الرياضية.

- عند استعمال التوابع الرياضية في أي برنامج بلغة C# يجب تضمين الصف Math والذي يحتوى على هذه التوابع.
- **مثل :**

Abs(x) ,Ceiling(x) , Cos(x) , Exp(x) , Floor(x) , Log(x) , Max(x) , Min(x),Pow(x), Sin(x) , Sgrt(x)

- **ويلخص الجدول التالي بعض التوابع الرياضية:**

Function	Description	Example
Sqrt(x)	الجذر التربيعي لـ x	sqrt (9.0) is 3
Pow(a,b)	الرفع إلى قوة	a^b pow(10,5)
Exp(x)	exp(1.0) is 2.718282	e^x
Abs(x)	القيمة المطلقة لـ x	if x > 0 fabs(x) = x = 0 fabs(x) = 0 < 0 fabs(x) = −x
Ceiling(x)	تقريب x لأصغر عدد صحيح أكبر من x	ceil(9.2) is 10.0 ceil(−9.8) is − 9.0
Floor(x)	تقريب x لأكبر عدد صحيح أصغر من x	floor(9.2) is 9 floor(−9.8) is −10.0
Max(x,y) Min(x,y)	ايجاد أكبر قيمة بين العددين x و y ايجاد أصغر قيمة بين العددين x و y	max(5,6)→ 6 min(7,10)→ 7



- البرنـامج التـالي يسـتخدم التوابـع الرياضـية ضـمن الصـف Math.
- يقوم بحساب :
  - القيمة المطلقة
  - الجذر التربيعي
  - الرفع إلى قوة
  - التوابع المثلية
  - التابع الأسي واللغارتمي

```
using System ;
class math1 {
public static void Main(string [ ] args ) {
 Console.WriteLine(" The pi number = "+Math.PI);
 Console.WriteLine(" Absolute of -15 is: = "+Math.Abs(-15));
 Console.WriteLine(" Sin(90) = "+Math.Sin((Math.PI)/2));
 Console.WriteLine(" Cos(90) = "+Math.Cos((Math.PI)/2));
 Console.WriteLine(" Smallest number = "+Math.Ceiling(9.3));
Console.WriteLine(" Rounding value of 10.5 is = "+Math.Round(10.5));
```

```
Console.WriteLine(" The Root Sguare of 36 = "+Math.Sqrt(36));
 Console.WriteLine(" The e number = " + Math.Exp(9)  );
 Console.WriteLine(" Largest Number = "+ Math.Floor(9.3));
 Console.WriteLine(" 2 Raised to the power of 4 = "+Math.Pow(2,4));
 Console.WriteLine(" log(1000) = "+Math.Log10(1000));
  } // end method main
} // end class Name
```

RESULT

The pi number = 3.141592653589793

Absolute of -15 is: = 15

Sin(90) = 1.0

Cos(90) = 6.123233995736766E-17

Smallest number(9.3) = 10.0

Rounding value of 10.5 is = 9

The Root Sguare of 36 = 6.0

The e number exp(9) = 8103.083927575384

Largest number(9.3) = 9

2 Raised to the power of 4 = 16.0

log(1000) = 3.0

Press any key to continue ……..

التوابع بدون وسطاء

Methods with Empty Parameter Lists

- في لغة C# يُكتب التابع الذي لا يمتلك وسطاء وذلك بترك القوسين فارغين ، فمثلاً الإعلان

```
public static void    print (  )  ;
```

- يشير إلى أن التابع print لا يأخذ أي وسيط وهو لا يرجع قيمة .
- والمثال التالي يبين الطريقة التي تكتب بها التوابع التي لا تأخذ وسطاء .
- **حيث التابع الأول يقوم بحساب وطباعة جداء عددين أما التابع الثاني يقوم بحساب وطباعة مجموع عددين .**




```
using System ;

class math1 {
   public static void   f1( ) {
   double x,y ;
       x=55;   y=44;
    Console.WriteLine( "x*y= {0}“, x*y) ;
   }
public static void  f2( ) {
   int x,y   ;
   x=5;y=7;
    Console.WriteLine("x+y= {0}“,(x+y)) ;
}
```

```
public static void Main(string [ ] args ) {
  f1 ( );
  f2 ( ) ;

 } // end method main
} // end class Name
```

RESULT

```
x*y = 2420
x+y = 12
Press any key to continue
```

التحميل الزائد للتوابع
Overloading Methods

- التحميل الزائد للتوابع يعنى استعمال الاسم نفسه لعدة توابع لكن كل تابع يجب أن يكون له تعريف مستقل. وعند استدعاء تابع يبحث المترجم عن نوع وسطاء التابع وعددها لمعرفة التابع المقصود. ولكي يميز المترجم بين تابع وآخر يحمل الاسم نفسه ، يقوم بعملية تعرف بتشويه الأسماء (names mangling)، تتألف هذه العملية من إنشاء اسم جديد خاص بالمترجم عن طريق دمج اسم التابع مع أنواع وسطائه.

مثال:

- البرنامج التالي يقوم بتحميل تابع square بشكل زائد لحساب الجذر التربيعي للنوع int وللنوع double .




```
using System ;

class Overloading {
   public static int square(int x)
{
   return   x*x ;
}
public static double square(double y)
{
    return    y*y ;
}
```

RESULT

The square of integer 7 is 49

The square of double 7.5 is 56.25

Press any key to continue

```
public static void Main(string [ ] args ) {
 Console.WriteLine( "The square of integer 7 is“ + square(7) ) ;
 Console.WriteLine( "The square of double 7.5  is "+ square(7.5) ) ;
  } // end method main
} // end class Name
```

- لنفرض لدينا برنامجاً يقوم بتحميل تابع بشكل زائد يدعى abs لحساب القيمة المطلقة لأعداد من النوع int ، double و long كما هو موضح في المثال (44) .




```
// abs is overloaded three ways
using System ;


class absOverloading {
public static void Main(string [ ] args ) {
 Console.WriteLine( abs (-10) );
 Console.WriteLine( abs(-11.0) );
 Console.WriteLine( abs( -9L ) );
} // end method main
public static int abs ( int  i ) {
Console.WriteLine("using integer abs( ) " );
  return    i<0 ? -i :i ;
}//End abs int
//Continued   ------------------
```

```
public static double abs ( double  d ) {
Console.WriteLine("using double abs( ) ");
  return   d<0.0 ? -d : d ;
}//End abs double
//Continued   --------------------
public static long abs( long  l ) {
Console.WriteLine("using long abs( ) ");
   return   l<0.0 ? -l : l ;
}//End abs Long
} // end class Name
```

RESULT

```
using integer abs( )
10
 using double abs( )
11
 using long abs( )
9
Press any key to continue
```

التمرير بالقيمة والتمرير بالعنوان

Pass by Value and Pass by Reference

- لنفرض أننا لدينا متحولين صحيحين في برنامج ونريد استدعاء تابع يقوم بتبديل قيمتي العددين ، ولنفرض أن العددين كما يلي:

```
int  x=1;
int  y=2;
```

1. التمرير بالقيمة(pass-by-value):

عند تمرير وسيط بطريقة الاستدعاء بالقيمة يجري إنشاء نسخة عن قيمة الوسيط لتمرر بعدها إلى التابع المستدعى وبالتالي لا تؤثر أية تغيرات تحدث على هذه النسخة على القيمة الأصلية للمتحول ضمن التابع المنفذ لعملية الاستدعاء.

تُرى هل يقوم التابع التالي بتبديل القيمتين:

```
public static void swap (int a, int b)
{
   int temp =a ;
   a=b ;
   b=temp ;
}
```

- يقوم هذا التابع بتبديل القيمتين a و b ، لكن إذا استدعينا هذا التابع كما يلي :

swap(x,y);

- سنجد أن قيمتي x و y لم تتغيرا , وذلك لأن الوسطاء العادية للتابع يتم تمريرها بالقيمة وينشئ التابع متغيرات جديدة كلياً عن a و b .

2. التمرير بالعنوان (pass-by-reference)

- عند تمرير وسيط بطريقة الاستدعاء بالمرجع فإن عنوان المتحول الأصلي هو الذي يتم تمريره إلى التابع وأية تغيرات على الوسيط ستؤثر على المتحول الأصلي
- التمرير بالعنوان(المرجع) هو طريقة تمكن التابع swap( ) من الوصول إلى المتحولات الأصلية x و y والتعامل معها بدلاً من إنشاء متحولات(نسخ) جديدة .
- ولإجبار تمرير الوسيط بالعنوان **نضيف ref** إلى نوع معطيات الوسيط في تعريف التابع وتصريح التابع.
- و يبين المثال (1) كيفية كتابة التابع swap وتمرير وسطائه بالعنوان.




```
using System ;

class byreference {
public static void Main(string [ ] args ) {
    int a=5 , b= 6 ;
   Console.WriteLine("sa= {0}“, sa(a)); //sa=25
   Console.WriteLine("  a= {0}“,  a ); // a not change , a=5
   Console.WriteLine("sb= {0}“, sb(ref b)); //sb=36
   Console.WriteLine("  b= {0}“,  b ); // b is change , b=36


} // end method main
```

```
public static int sa ( int x )
{
   return  x = x*x ;
}
public static int sb ( ref int y )
{
   return y =y*y;
}
} // end class Name
```

RESULT

```
 sa=25          a = 5
sb=36           b = 36

Press any key to continue
```

- يمكن استخدام الكلمة المفتاحية out بدلاً من ref .
- استخدام out مشابه بشكل كبير لاستخدام ref ، لكن الكلمة المفتاحية out تُعلم المترجم أن التابع الذي يتم تمرير المتحول إليه سيقوم بمهمة أعطاء قيمة ابتدائية له .
- لأنه عند تمرير متحول بالعنوان باستخدام الكلمة المفتاحية ref يجب إسناد قيمة ابتدائية للمتحول الذي يتم تمريره للتابع .
- سيتم توضيحه في فصل الصفوف عند إنشاء غرض من صف .

تابع العودية
Recursion Method

- من الميزات التي ظهرت في اللغات الحديثة هو تابع العودية (التكرار والتوليد) .
- تتلخص هذه الميزة بإمكانية التابع أن يستدعي نفسه , وأن الفائدة التي يجنيها المبرمج من استدعاء التابع نفسه باستعمال تابع العودية , هو أن هذا التابع يستخدم لحل بعض المسائل المرتبة ترتيباً متكرراً أصلاً كمسائل ترتيب الأرقام تصاعدياً أو تنازلياً في برنامج quicksort .
- كما أن هذا النوع من التوابع يفيد في برمجة بعض مسائل الذكاء الصنعي AI(Artificial Intelligent) .
- عند استخدام تابع العودية نحتاج أن نستعمل التعليمة الشرطية if لإجبار التابع للعودة إلى مكان انطلاقه بعد استدعاء نفسه في البرنامج, وإن لم نستعمل التعليمة if فإن التابع لا يعود .

مثال

- مثال لإيجاد مجموع n عدد صحيح مرتب باستخدام مفهوم العودية ,
لنفرض لدينا التابع التالي :

```
public static int   sum( int  n ) {
   if ( n  = =  1 )
         return 1;
   else
         return  n + sum(n-1) ;
}
```

- هذا التابع يعني أن **Sum(n) = n + sum(n-1)** كما يلي :

- عند استدعاء هذا التابع على شكل sum(5) , فإن الحاسوب يقوم عند الاستدعاء الأول بتنفيذ 5+sum(4) فإن التعليمة return تقوم باستدعاء التابع sum(4) مرة أخرى .
- وأما التابع sum(5) الذي استدعي في المرة الأولى فسوف يحتفظ بنسخة منه في الذاكرة على شكل تابع غير منتهي .
- وبنفس الطريقة فإن التابع sum(4) الذي استدعي في المرة الثانية سيبقى غير منتهي , و هكذا حتى يتم استدعاء sum(1) .
- نلاحظ أن الشرط في sum(1) قد تحقق وأن sum(1) سينفذ التعليمة return (1) التي هي القيمة الذي ينتظرها التابع sum(2) والذي سينفذ التعليمة return وهكذا وحتى نصل إلى sum(5) .

• يبين المثال التالي حساب n! (n عاملي) وأيضا حساب مجموع n عدد صحيح (من 1 حتى n) .

طريقة التنفيذ لحساب مجموع الأعداد من 1 حتى 5

n = 5 ➡ sum (5)= 5+ sum (4)

n = 4 ➡ sum (4) = 4 + sum (3)

n = 3 ➡ sum (3)= 3 + sum (2)

n = 2 ➡ sum (2)= 2 + sum (1)

n = 1 ➡ sum (1)= 1 ⇨ 1 ⇨ 3 ⇨ 6 ⇨ 10 ⇨ 15

طريقة التنفيذ لحساب !5

n = 5 → Fact (5)= 5* Fact (4)

n = 4 → Fact (4) = 4 * Fact (3)

n = 3 → Fact (3)= 3 * Fact (2)

n = 2 → Fact (2)= 2 * Fact (1)

n = 1 → Fact (1)= 1 ⇒ 1

⇒ 2 ⇒ 6 ⇒ 24 ⇒ 120




```
using System ;

class Recursion {
public static void Main(String [ ] args ) {
   int n = 5 ;

Console.WriteLine( "Factorial "+n+" is "+fact(n) );
Console.WriteLine("Summing "+n+" is"+sum(n) );

} // end method main
```

```
public static int  fact( int  n) {
    if (n == 0 || n = =  1 )
          return 1;
    else
          return  n * fact(n-1);
}//End fact

public static int  sum( int  n) {
    if (n == 1)
          return 1;
    else
          return  n + sum(n-1);
}//End sum
} // end class Name
```

RESULT

```
Factorial   5  is  120
Summing  5  is  15

Press any key to continue
```

توليد الأرقام العشوائية
Random Number Generation

- جميع لغات البرمجة توفر توابع أو طرائق أو صفوف تقوم على إنشاء أرقام عشوائية Random وقد تستخدم هذه الارقام العشوائية في كثير من الامور وأهمها ما يتعلق بالألعاب فمثلاً في العاب الأوراق (الشدة) كل مرة تأتي الاوراق مختلفة أي عشوائية Random، ولعبة النرد حيث لحجر النرد ستة وجوه وكثير من الامور تحتاج إلى أرقام عشوائية .
- **لغة C# وفرت صف يقوم على إنشاء الأرقام العشوائية يسمى Random، التابع المسؤول عن إنشاء الأرقام العشوائية يسمى Next() وهو غير معرف على انه static لذلك يتوجب علينا أولاً إنشاء غرض من الصف Random ثم الوصول اليه عن طريق الغرض.**
- التابع Next() هو يقبل التحميل الزائد Overloading إذا قمنا باستدعائه بدون ارسال أي براميتر يرجع لنا رقماً عشوائياً Random Number من 0 حتى آخر رقم موجود في لغة C# هو Max Number أي (0 إلى 2,147,483,647) .

- يتم إنشاء غرض من الصف Random كما يلي :

```
Random rnd = new Random()
```

- التابع Next() هو المسؤول عن توليد الأرقام العشوائية ، وله وسيطان : الوسيط الأول هو الحد الأدنى وهو يمثل القيمة الابتدائية والوسيط الثاني هو الحد الأعلى وهو يمثل القيمة العظمى أي Next(min,Max+1) .
- مثلاً لو أردنا توليد عدد عشوائي بين 1 و 12 :

```
rnd.Next(1,13) ;
```

حيث يولد التابع رقماً عشوائياً بين 1 و 12 .

- التابع Next() يعيد قيمة فإن استدعائه بهذه الطريقة خطأ ، حيث يجب اسناده إلى متغير من نوع القيمة المعادة ، أي :

```
int rNum ;
rNum = rnd.Next(1,13) ;
```



Code	Expected output
`Random x = new Random();` `Console.WriteLine(x.Next());`	**0 - 2,147,483,647**
`Random x = new Random();` `Console.WriteLine(x.Next(5));`	0 - 4
`Random x = new Random();` `Console.WriteLine(x.Next(1,5));`	1 - 4

مثال(46-a)

- **المثال التالي يوضح :**

✓ توليد رقم عشوائي من 1 حتى 7 يدل على أيام الأسبوع .

✓ توليد رقم عشوائي من 1 حتى 12 يدل على اسم الشهر .

✓ بينما العام يبقى ثابت على 2017 .

```
using System;
class RandIntegers {
    public static void Main(string[] args){
// random number generator
        Random rnd = new Random();
        Random rnd1 = new Random();
        int rNum;
         int rNum1;
        rNum = rnd.Next(1, 8);
        rNum1 = rnd1.Next(1, 13);
  DateTime dt = Convert.ToDateTime(rNum + "/" + rNum1 + "/ 2017");
  Console.WriteLine(dt); // display generated value
  Console.WriteLine(dt.ToString("MMMM")); // display generated value
switch (rNum) {
 case 1:
     Console.WriteLine("Sunday"); break;
```

```
case 2:
   Console.WriteLine("Monday");
   break;
case 3:
   Console.WriteLine("Tuesday");
   break;
case 4:
   Console.WriteLine("Wednesday");
   break;
case 5:
   Console.WriteLine("Thursday");
   break;
case 6:
   Console.WriteLine("Friday");
   break;
```

```
case 7:
  Console.WriteLine("Saturday");
  break;
default:
  Console.WriteLine("Sorry, we are out");
  break;
        }//end switch
    } // end method main
} // end class Name
```

RESULT

```
  04/06/2017 00:00:00
June
Wednesday
-----------------------
06/10/2017 00:00:00
October
Friday
Press any key to continue
```

- على سبيل المثال في برنامج يحاكي عملية رمي قطعة نقود فنحن نحتاج إلى قيمتين فقط هما : 0 من أجل الطرة head , و1من أجل النقش tail .
- أما بالنسبة لبرنامج يحاكي عملية رمي حجر النرد ذو ستة وجوه , فإننا نحتاج إلى ستة قيم ضمن المجال من 1 حتى 6 .
- حتى نبين كيفية عمل الصف Random , لنقم بكتابة برنامج يحاكي عملية رمي نرد ذو ستة وجوه 20 مرة حيث يتم طباعة نتيجة كل عملية من عمليات الرمي .
-

مثال (46)

```
// Shifted, scaled integers produced by 1 + rand( ) % 6
using System ;

class RandIntegers {
public static void Main( string [ ] args )
  {                                        // random number generator
     Random   r = new  Random( ) ;
      int face; // stores each random integer generated loop 20 times
   for ( int counter = 1; counter <= 20; counter++ )
    {   // pick random integer from 1 to 6
      face = r.Next (1, 7 );
```

```
Console.Write ( "   {0}", face ); // display generated value

  // if counter is divisible by 5, start a new line of output
        if ( counter % 5 == 0 )
        Console.WriteLine( );
      } // end for

    } // end method main
} // end class Name
```



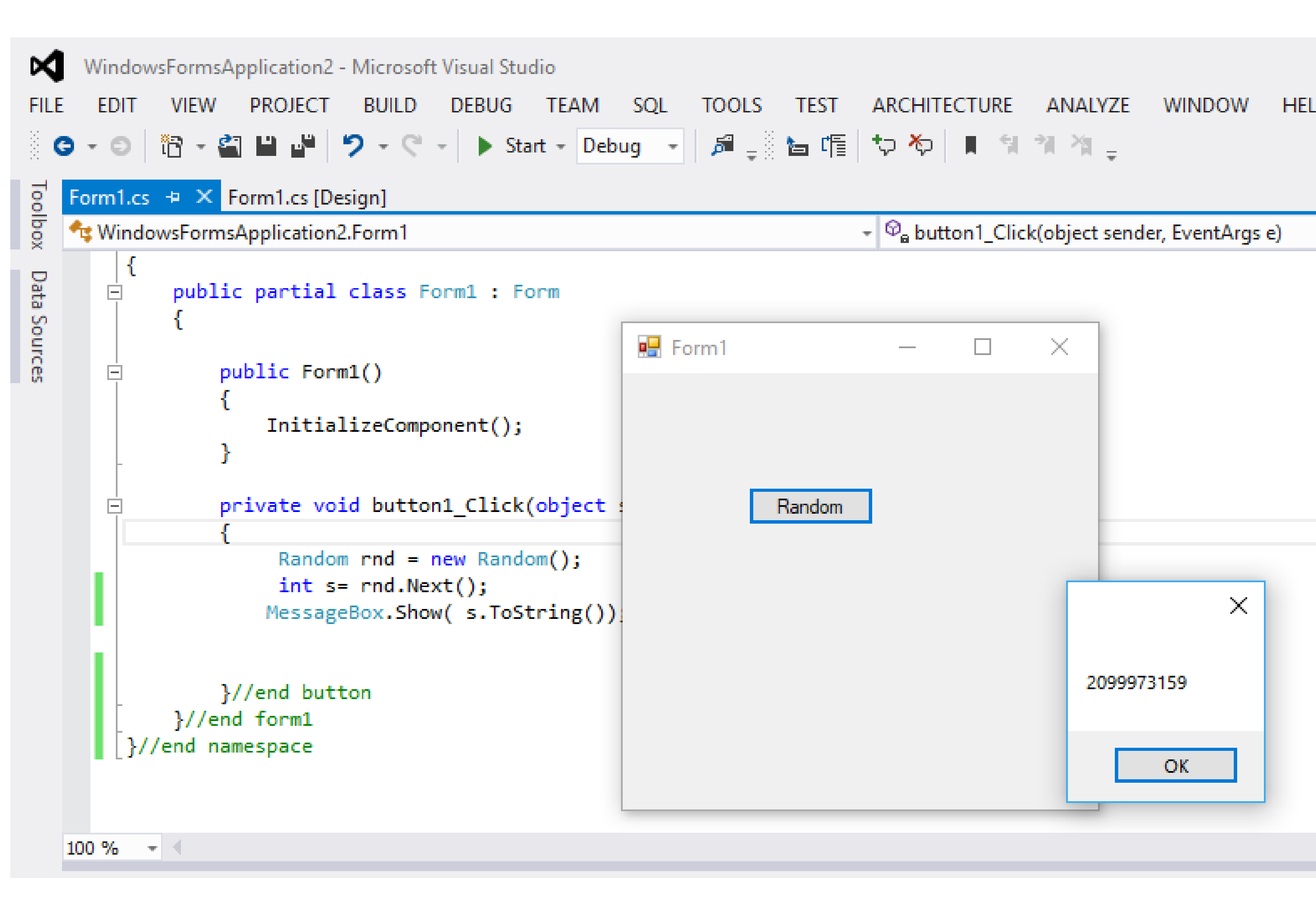
مثال(46-b)

- البرنامج (a) في المثال التالي يوضح توليد الأرقام العشوائية باستخدام تطبيقات ويندوز .
- بينما البرنامج (b) في المثال التالي يوضح توليد الأرقام العشوائية باستخدام تطبيقات ويندوز وتدل على الشهر .

البرنامج(a)

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication2 {
    public partial class Form1 : Form {

        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
             Random rnd = new Random();
             int s;
             s = rnd.Next();
             MessageBox.Show( s.ToString());
        }//end button
    }//end form1
}//end namespace
```


WindowsFormsApplication2 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW
Start
Debug
Form1.cs
Form1.cs [Design]
WindowsFormsApplication2.Form1
button1_Click(object sender, EventArgs e)
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
}
private void button1_Click(object
{
Random rnd = new Random();
int s= rnd.Next();
MessageBox.Show(s.ToString());
}//end button
}//end form1
}//end namespace
Toolbox
Data Sources
100 %
Form1
Random
2099973159
OK

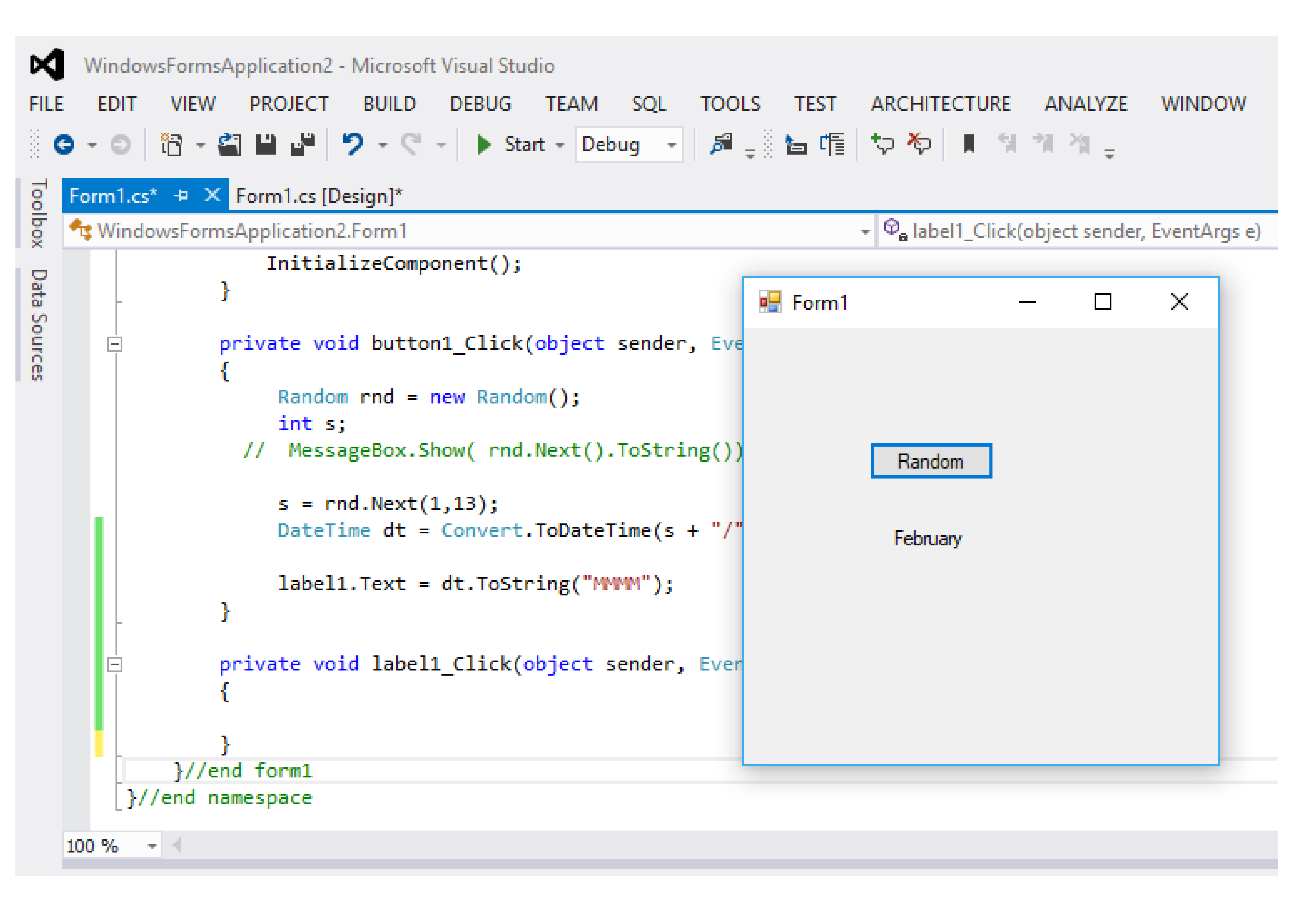
البرنامج(b)

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication2 {
  public partial class Form1 : Form {
     public Form1()
      {
        InitializeComponent();
       }
    private void button1_Click(object sender, EventArgs e)
     {
       Random rnd = new Random();
      int s;
      s = rnd.Next(1,13);
     DateTime dt = Convert.ToDateTime(s + "/" +  s + "/ 2017");
    label1.Text = dt.ToString("MMMM");
 }//end button
```

```
private void label1_Click(object sender, EventArgs e)
        {

        }

    }//end form1
}//end namespace
```



WindowsFormsApplication2 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW
Start
Debug
Form1.cs*
Form1.cs [Design]*
Toolbox
Data Sources
WindowsFormsApplication2.Form1
label1_Click(object sender, EventArgs e)
InitializeComponent();
}
private void button1_Click(object sender, Eve
{
Random rnd = new Random();
int s;
// MessageBox.Show(rnd.Next().ToString())
s = rnd.Next(1,13);
DateTime dt = Convert.ToDateTime(s + "/"
label1.Text = dt.ToString("MMMM");
}
private void label1_Click(object sender, Even
{
}
}//end form1
}//end namespace
100 %
Form1
Random
February

تمارين عامة(2)

1. اكتب برنامجاً بلغة #C يحتوي على تابعين : التابع الأول اسمه Factorial مهمته حساب العاملي (!) لأي عدد صحيح موجب , أما الثاني اسمه SumFactorial مهمته حساب المقدار التالي :

$$sum = n! + \frac{n!}{x!} - m!$$

التابع main يستدعي كلاً من التابعين وطباعة قيمة المقدار Sum (التابع SumFact يستدعي التابع Factorial) .

=========================

2. اكتب برنامجاً بلغة #C يقوم بما يلي : إدخال قيم المتحولات a,b,x وهي من النمط double ويحتوي البرنامج على تابعين :

– التابع الأول اسمه FunctionY مهمته حساب المقدار التالي :

$$y = x^2 + 2x + 10$$

– أما التابع الثاني اسمه FunctionZ مهمته حساب المقدار Z والمعرف كما يلي :

$$Z = (y - a)^2 + (y + b)^2$$

– التابع main يستدعي كلاً من التابعين لطباعة قيمة Z (التابع FunctionZ يستدعي التابع FunctionY) .

==============================

3. اكتب برنامجاً بلغة C# يقوم بإدخال أي عدد حقيقي x من قبل المستخدم .ويحتوي على ثلاثة توابع لحساب قيمة التابع الرياضي التالي :

$$yy = \begin{cases} 5x^5 + 2x + 5 \text{.....} if \text{..} 1 \le x < 5 \\ \dfrac{x^6}{2x+1} \text{............} if \text{...} 5 \le x < 100 \\ 3x + 5 \text{.............} otherwise \end{cases}$$

التـابع main يستدعي كـلاً مـن التوابـع الثلاثـة لطباعـة قيمـة yy فـي التابع main.

==============================

4. أكتب برنامجاً بلغة C# يقوم بما يلي:

– إدخال أي عدد صحيح x.
– يستخدم تابعاً اسمه fun_prim لإيجاد قواسم العدد المدخل x .
– يختبر البرنامج إذا كان العدد المدخل x أولياً أو غير أولياً وذلك بناءً على خرج التابع fun_prim, وطباعة النتيجة في التابع main

5. أكتب برنامجاً بلغة #C يقوم بما يلي:

- إدخال أي عددين صحيحين p و k .
- يستخدم تابعين :
- التابع الأول اسمه Factorial لحساب قيمة !p (العاملي)
- التابع الثاني اسمه F_permutation لحساب قيمة المقدار الرياضي التالي:

$$permutation = \frac{p!}{(p-k)!}$$

- التابع main يستدعي كلاً من التابعين لطباعة قيمة Permutation وطباعة النتيجة في التابع main.



المصفوفات

Arrays

Chapter 4

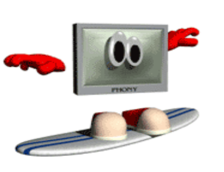
المصفوفات
Arrays



- المصفوفات أحادية البعد
- المصفوفات متعددة الأبعاد

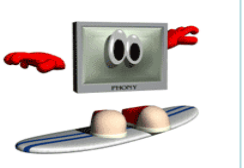
محتويات الفصل الثاني

- لمصفوفات أحادية البعد Arrays
  - التصريح عن المصفوفة
  - ادخال وطباعة المصفوفة
  - تمرير المصفوفات كوسطاء للتوابع
  - أمثلة وتمارين
- المصفوفات ثنائية البعد Tow Dimension Arrays
  - التصريح عن المصفوفة
  - ادخال وطباعة المصفوفة
  - تمرير المصفوفات كوسطاء للتوابع
  - أمثلة وتمارين

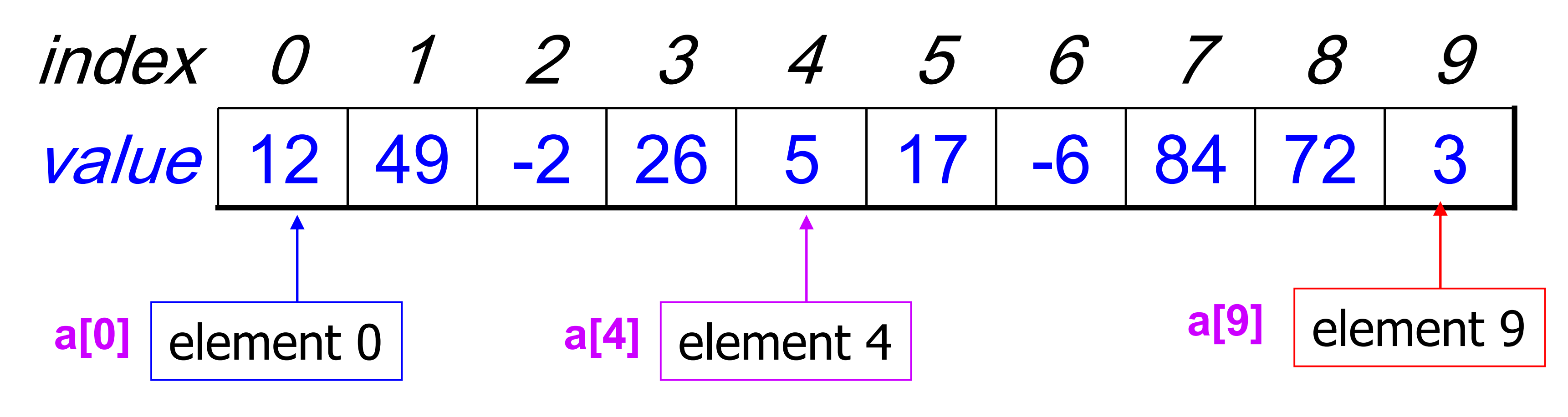


1- المصفوفات أحادية البعد
One Dimensional Arrays

- المصفوفة هي بنية تُستخدم للاحتفاظ بمجموعة من القيم من نفس نمط المعطيات .
- المصفوفة هي نوع من أنواع بنى المعطيات، لها عدد محدود ومرتب من العناصر التي تكون جميعها من نفس النوع type، فمثلاً يمكن أن تكون جميعها صحيحة int أو حقيقية floatأو محرفية char , ولكن لا يمكن الجمع بين نوعين مختلفين في نفس المصفوفة .
- الشكل التالي يبين مصفوفة a تحتوى على 10 عناصر من النوع int، ويمكن الوصول إلى أي من هذه العناصر بذكر اسم المصفوفة متبوعاً برقم موقع العنصر في المصفوفة محاطاً بالأقواس [ ] ، وبالتالي ، إن a[10] مصفوفة اسمها a عدد عناصرها 10 .



- يرمز لرقم العنصر في المصفوفة بدليل العنصر index .
- دليل العنصر الأول في المصفوفة هو 0 ولهذا يشار إلى العنصر الأول في المصفوفة a بـ a[0] والثاني a[1] والسابع a[6] وعموماً يحمل العنصر i في المصفوفة a الدليل a[i−1] .
- وتتبع تسمية المصفوفات نفس قواعد تسمية المتحولات.
- **مثال : لدينا مصفوفة مكونة من 10عناصر من نمط int**



1- التصريح عن متحول يشير إلى مصفوفة :

- تحتل المصفوفات حجماً في الذاكرة لذا يجب على المبرمج تحديد نوع عناصر المصفوفة وعددها حتى يتسنى للمعرف تخصيص الحجم اللازم من الذاكرة لحفظ المصفوفة.
- إن التصريح عن مصفوفة يشبه التصريح عن أي متحول آخر , فالتصريح يحوي قسمين :
  - الأول هو نوع المصفوفة
  - والثاني هو اسم المصفوفة .
- يتم كتابة نوع المصفوفة كما يلي :

Type [ ] anArray

حيث Type هو نمط المعطيات , وأما القوسين [ ] فيشيران إلى أن هذا المتحول هو مصفوفة وليس قيمة واحدة .

- **مثال :** string [ ] anArray1 , int [ ] anArray2 , ...

- إن التصريح عن متحول مصفوفة لن يحجز أي منطقة في الذاكرة حتى يتم فعلياً إسناد قيم إلى المصفوفة .

2 -إنشاء مصفوفة

- يتم إنشاء مصفوفة باستخدام المعامل **new** .
- **مثلاً :** anArray = **new** int [20] ;
  – هذه العبارة تقوم بإنشاء مصفوفة وتخصيص ذاكرة محددة لها تتسع لعشرين عدداً صحيحاً .

anArray = **new** double [20] ;

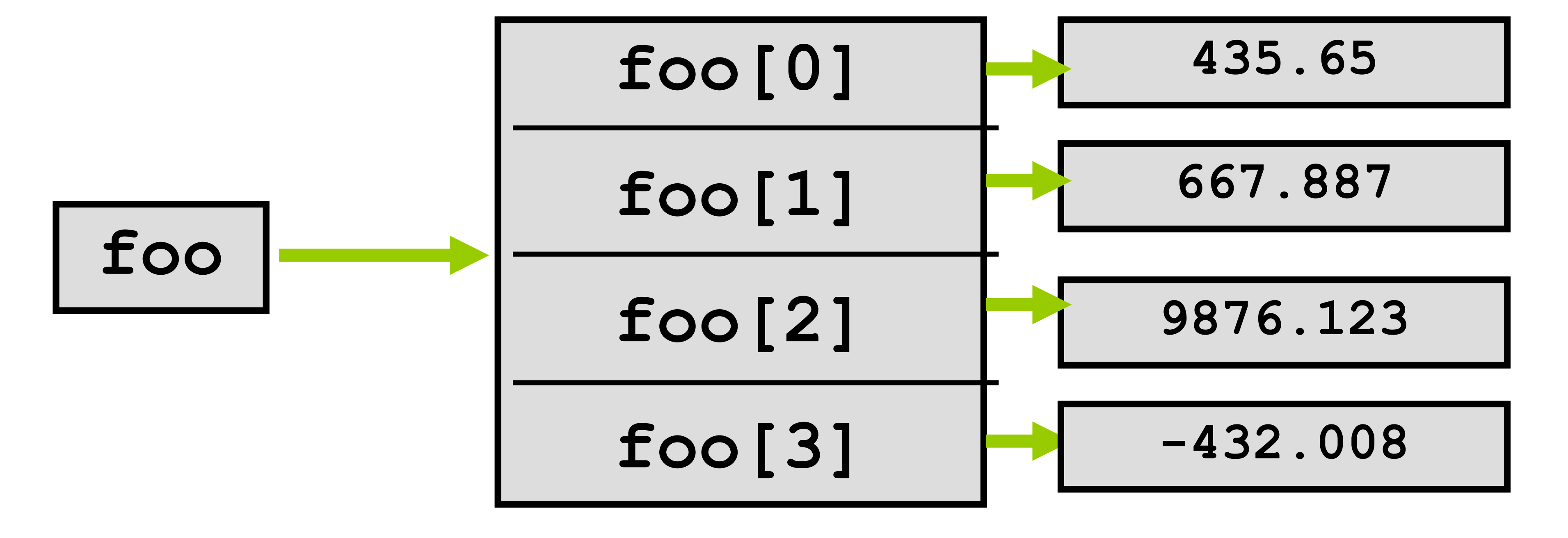
– إنشاء مصفوفة وتخصيص ذاكرة محددة لعشرين عدداً حقيقيا

anArray = **new** string [20] ;

– بإنشاء مصفوفة وتخصيص ذاكرة محددة لعشرين سلسلة

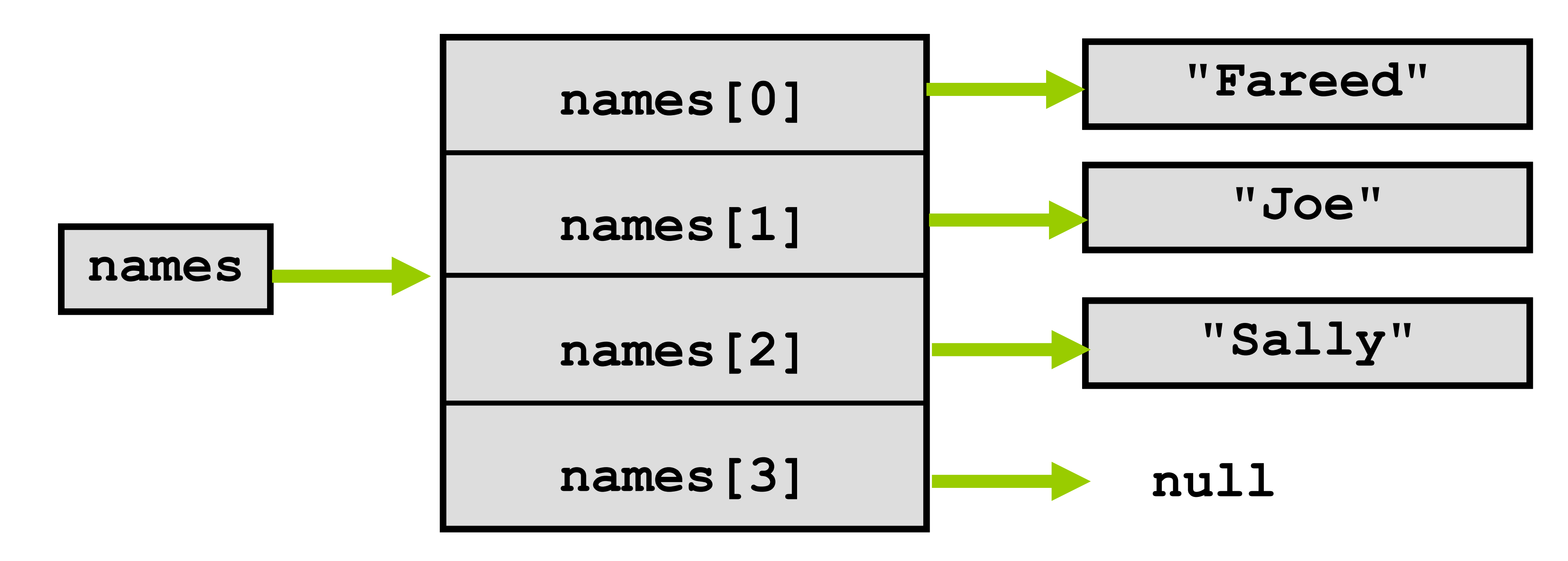
- التصريح عن مصفوفة foo من نمط double تتكون من 4 عناصر (إنشاء مصفوفة) .

```
double []foo ;
foo = new double [4] ;
```



- التصريح عن مصفوفة **names** من نمط string تتكون من 4 عناصر (إنشاء مصفوفة) .

string [ ] names = new String [4];



الحصول على طول المصفوفة

- للحصول على طول المصفوفة (عدد عناصرها) نستخدم الصيغة التالية :

Arrayname.Length

- تعيد العبارة Arrayname.Length حجم المصفوفة
- مثلاً :

```
for (int j=0; j<array.length ; j++)
    {
            For body
    }
```



إدخال وإخراج المصفوفة

- يمكن إعطاء قيمة ابتدائية لعناصر المصفوفة بإتباع التصريح عن المصفوفة بعلامة المساواة (=) تليها لائحة من القيم المطلوب إسنادها لعناصر المصفوفة عندها، ويتم الفصل بين القيم بفواصل، وتحيط هذه اللائحة الأقواس الحاصرة { }.
- **مثال :**

```
int [ ]n = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37};
int [ ]n =int[10] { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37}
```
أو

- و يقوم البرنامج التالي بإسناد عناصر من النوع integer لتحتوي قيم محددة عند التصريح عن المصفوفة، وطباعة هذه القيم.




```
//initializing an array
using System ;
 namespace   Example48 {
class ArrayProg {
public static void Main(string  [ ] args ) {
int [ ]arry = {  32 ,27,64,18,95,14,90,70,60,37} ; // initialize array
 // int [ ]arry =int[10] {  32 ,27,64,18,95,14,90,70,60,37} ;
for (int  i=0 ; i< 10; i++)   // print array
Console.WriteLine( i+"   " +arry[i]) ;
} // end method main
} // end class Name
 } //end namespace
```

RESULT

```
0    32
1    27
2    64
3    18
4    95
5    14
6    90
7    70
8    60
9    37
Press any key to continue
```

ماذا يحدث إذا تم تحديد حجم مصفوفة لا يتوافق مع عدد القيم الابتدائية الموجودة في اللائحة؟



- يستخدم البرنامج التالي حلقة for ليقوم :
  - بتصفير عناصر المصفوفة n و طباعتها
  - بإدخال عناصر المصفوفة من المستخدم وطباعتها

```
using System ;
 namespace   Example49 {
class IOArry {

public static void Main(string [ ] args ) {
int   [ ]n;

n = new int [10] ;
  for (int i=0; i<10;i++) // initialize array
          n[i] = 0;
```

```
for (int i=0 ; i< 10; i++)   // print array
Console.WriteLine( i+"  " +n[i]) ;
for (int i=0 ; i< n.Length; i++ )
//Enter new elements
n[i] = int.Parse( Console.ReadLine() );
for (int i=0 ; i< n.Length; i++)
Console.WriteLine( i+"  " +n[i]) ;// print array

} // end method main
} // end class Name
 } //end namespace
```

RESULT

0	0	0	11
1	0	1	22
2	0	2	33
3	0	3	44
4	0	4	55
5	0	5	66
6	0	6	77
7	0	7	88
8	0	8	99
9	0	9	100

11 22 33 44 55 66 77 88 99 100

- **ملاحظات هامة**

1. إذا كانت القيم الابتدائية الموجودة في اللائحة أكثر من حجم المصفوفة المحددة فإن المترجم سيولد خطأ، وإذا كانت أقل سيملأ المترجم بقية العناصر أصفاراً.
2. لذا إذا كان المطلوب إسناد عناصر مصفوفة مهما كان حجمها بأصفار , يمكن التصريح عن المصفوفة بالشكل التالي :

```
int  anyarray[10]={0};
```

3. سيتم إسناد القيمة 0 للعنصر الأول أما العناصر المتبقية يسند لها القيمة صفر أيضاً كوننا لم نحدد قيمة لها.
4. يقوم البرنامج التالي بجمع 12 عنصراً في مصفوفة من النوع int .




```
// compute the sum of the elements of the array
using System ;

class ArrayProg {
public static void Main(string [ ] args ) {
const  int size =12;
int [ ] a = new int [size ] {1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45};
int total = 0;
for (int i= 0; i< a.Length ; i++)
        total = total+ a[i];
Console.WriteLine(" total of array element values is "+  total );

} // end method main
} // end class Name
```





- نلاحظ في العبارة:

```
const int arraysize = 12;
```

- استعملنا كلمة جديدة هي const . يتم استعمال هذه الكلمة الأساسية في تعريف المتحول الذي لا يمكن تغيير قيمته(الثابت) في البرنامج ولذلك يجب إعطاءه قيمة ابتدائية عند تعريفه (في البرنامج السابق تم تحديده بالقيمة 12)
- يبين البرنامج التالي إدخال عناصر المصفوفة وطباعة عدد أيام الشهر (28 أو 30 أو 31) وفي مثالنا الشهر رقم (4) هو الشهر **April** **ويقابله المصفوفة** month days[3] .








```
// Demonstrate a one-dimensional array.
using System ;
 namespace   Example51 {
class Array {
public static void Main(string  [ ] args) {
int [ ]month_days;
//int [ ] month_days = { 31, 28, 31, 30, 31, 30, 31,
                                 31, 30, 31, 30, 31 };
month_days = new int[12];
month_days[0] = 31;
month_days[1] = 28;
month_days[2] = 31;
```

```
month_days[3] = 30;
month_days[4] = 31;
month_days[5] = 30;
month_days[6] = 31;
month_days[7] = 31;
month_days[8] = 30;
month_days[9] = 31;
month_days[10] = 30;
month_days[11] = 31;
Console.WriteLine("April has " + month_days[3] + " days.");
} // end method main
} // end class Name
 } //end namespace
```

RESULT

April has 30 days.

Press any key to continue

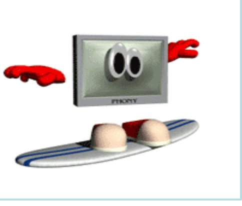


البرنامج التالي يقوم بحساب المتوسط
الحسابي لعناصر المصفوفة




```
// Average an array of values.
using System ;
 namespace   Example52 {
class Average {
public static void Main(string [ ] args) {
double [ ] nums = {10.1, 11.2, 12.3, 13.4, 14.5};
double result = 0;
for(int i=0; i<nums.Length; i++)
result = result + nums[i];
Console.WriteLine("Average is " + result / 5);
} // end method main
} // end class Name
 } //end namespace
```







- **لدينا 30 طالباً يدرسون مقرر البرمجة والمطلوب : اكتب برنامجاً بلغة ++C يقوم بما يلي :**
  - إدخال درجة الطالب d وطباعة كلمة ناجح" Passed " إذا كانت درجة الطالب 60 ≤ d ويطبع اسمه Name ودرجته وكلمة راسب " Failed " إذا كانت غير ذلك .
- ارسم المخطط التدفقي اللازم .
- استخدم الحلقة **while , for**

```
using System ;
 namespace   Example53 {
class Name {
public static void Main(string [ ] args) {
  double  d;
string   Name;
for(int  i =0 ;i < 30; i++) {
  Console.WriteLine(" degree & Name ");
d = double.Parse( Console.ReadLine() );
Name = Console.ReadLine()  ;
if ( d >= 60 )
 Console.WriteLine(" Passed = {0}   {1}“,d,Name);
else
 Console.WriteLine (" Failed = {0} “,d);
  }//End for
} // end method main
} // end class Name
 } //end namespace
```

Start
I=1
Input d,Name
d >=60
No
Yes
Print
d راسب
Print ناجح
d ,Name
I = I + 1
I <= 30
Yes
No
END



- لقد طبقنا البرنامج بالنسبة لأربعة طلاب فقط

RESULT

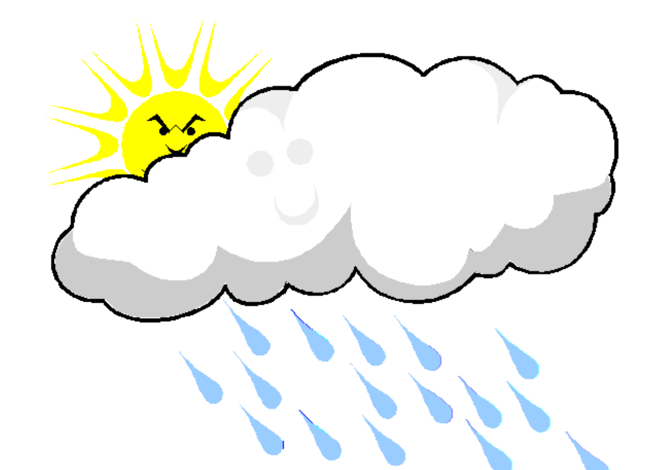
```
degree & Name
80  Waseem
 Passed = 80   Waseem

 degree & Name
90  Ali
 Passed = 90   Ali

 degree & Name
55 Omar
 Failed = 55

 degree & Name
70  Younes
 Passed = 70   Younes

Press any key to continue
```



تمرير المصفوفات كوسطاء للتوابع

- عند تمرير مصفوفة على شكل وسيط لتابع عند استدعائه في التابع main( ) يجب استخدام اسمها فقط بدون أقواس , فعلى سبيل المثال , إذا تم التصريح عن المصفوفة a على الشكل التالي :

int [ ] a = new int[20] ;

– فإن الاستدعاء التالي للتابع sumArray:

sumArray(a , 20) ;

– يأخذ التابع اسم المصفوفة وحجمها كوسيطين له .

- **عند تمرير مصفوفة إلى تابع فإنه يتم تمرير حجمها أيضاً حتى يتعامل مع العدد المحدد من عناصر المصفوفة الممررة .**

- استقبال المصفوفة المرسلة عند استدعاء التابع فيجب أن تحتوي قائمة وسطاء التابع عند تعريفه على اسم المصفوفة مع الأقواس [ ] a , ويجب كتابة التابع sumArray على الشكل التالي :

Public Static void sumArray(int a[ ] , int size) { }

- للدلالة على أن التابع sumArray يتوقع استقبال مصفوفة من الأعداد الصحيحة حسب الوسيط a وهي تحتوي على 10 عناصر .
- ليس من الضروري وضع حجم المصفوفة بين القوسين عند كتابة الوسيط [a[20 , وفي حال القيام بذلك فإن المترجم يقوم بتجاهله .
- وعلى اعتبار أن المصفوفات يتم تمريرها بالعنوان فإن التابع المُستدعى يقوم باستخدام اسم المصفوفة a من أجل الرجوع إلى المصفوفة الأصلية الموجودة لدى التابع الذي تم الاستدعاء ضمنه المصفوفة .
- يبين المثال التالي تمرير المصفوفة للتابع **sumArray** ليقوم بحساب مجموع عناصر المصفوفة وطباعته .

مثال(54)

```
// compute the sum of the elements of the array
using System ;
 namespace   Example54 {
class PassArray {
public static void Main(string [ ] args ) {
int [ ] arry  = new int [10] ;
for ( int i=0 ; i< arry.Length ; i++ ) {
Console.WriteLine(" Enter Array  ");
int x = int.Parse( Console.ReadLine() );
   arry[i]=x;
  } // End for
   sumArray ( arry, 10 ) ;
 } // end method main
```

RESULT

11 22 33 44 55 66 77 88 99 100
total of array element values is
595
Press any key to continue

```
public static void  sumArray ( int [ ] a , int  n) {
int total = 0;
for (int i= 0; i< n ; i++)
        total = total+ a[i];
  Console.WriteLine(" total of array element values is "+ total ) ;
}//End function
} // end class Name
 } //end namespace
```

- يوجد العديد من الحالات التي لا نرغب فيها بتغيير قيم عناصر المصفوفة .
- وعلى اعتبار أن عملية تمرير مصفوفة تتم بالعنوان فإنه من الصعب التحكم بعملية التغيير .
- **لكن يتوفر في C# التعليمة const لمنع القيام بأي تغيير على قيم عناصر المصفوفة الممررة .**
- عند استخدام التعليمة const من قبل أي مصفوفة معطاة كوسيط فإن عناصرها تصبح عبارة عن قيم ثابتة ضمن جسم التابع , ولا يمكن تغيير قيمها وأي محاولة لذلك ستؤدي إلى حدوث خطأ أثناء ترجمة البرنامج , مما يساعد المبرمج على تصحيح برنامجه بشكل لا يسمح بالقيام بعمليات التغيير غير المطلوبة .
- نعيد كتابة المثال السابق باستخدام التعليمة const مع التابع sumArray .




```
// compute the sum of the elements of the array
using System ;
 namespace   Example55 {
class PassArray {
public static void Main(string [ ] args ) {
int [ ] arry  ;
const  int size = 10;
arry = new int [sze] ;
for ( int i=0 ; i< size ; i++ ) {
Console.WriteLine(" Enter Array  ");
int x = int.Parse( Console.ReadLine() );
   arry[i]=x;
  } // End for
   sumArray ( arry, 10 ) ;
 } // end method main
```

```
public static void  sumArray ( int [ ] a , int  n) {
int total = 0;
for (int i= 0; i< n ; i++)
        total = total+ a[i];
  Console.WriteLine(" total of array element values is "+ total ) ;
}//End function
} // end class Name
 } //end namespace
```

RESULT

11 22 33 44 55 66 77 88 99 100
total of array element values is
595
Press any key to continue

الحلقة foreach

- تُقدم لغة C# نموذجاً خاصاً من حلقة for هي الحلقة foreach تُتيح لنا الوصول إلى جميع عناصر المصفوفة .
- حيث تقوم حلقة foreach بالتكرار عبر عناصر المصفوفة بدون استخدام عداد ، بحيث لا يمكن إضافة أو إزالة العناصر ضمن الحلقة، أي أن حالة المصفوفة يجب أن تبقى كما هي بدون تغيير أثناء الإعادة والتكرار عبر عناصرها .
- **وتتميز الحلقة foreach بما يلي :**
  - ➢ أن الحلقة foreach لا تحتاج لعداد.
  - ➢ الحلقة foreach ليس لديها شرط للبداية .
  - ➢ الحلقة foreach لا تتوقف حتى تعرض كل العناصر.
  - ➢ الحلقة foreach لا تحتاج أن نعطيها كم عدد العناصر التي ستتعامل معها.

```
string[ ] weekDays =new string[4] { "Sunday", "Monday", "Tuesday"};

foreach (string day in weekDays)
  {
      System.Console.WriteLine("The day is : {0} ",day);
   }
```

- تم إنشاء متغير اسمه day من نوع string) لأن المصفوفة weekDays تحتوي على أسماء الأيام أي هي من نوع string) ، تعني in أي في المصفوفة weekDays. .
- هذا معناه من أجل كل عنصر نصي Day من المصفوفة weekDays قم بالعملية التالية :

```
System.Console.WriteLine("The day is : {0} ",day);
```

- البرنامج التالي يوضح كيفية طباعة عناصر المصفوفة بواسطة حلقة foreach .

مثال(55-a)

```
using System;
class ArrayProg
 {
     public static void Main(string[] args)
     {
       string[] weekDays = new string[7] { "Sunday", "Monday",
   "Tuesday","Wednesday", "Thursday", "Friday", "Saturday" };

    foreach (string day in weekDays) {
     System.Console.WriteLine("The day is : {0} ", day);
        }  // end foreach


        } // end method main
    } // end class Name
```

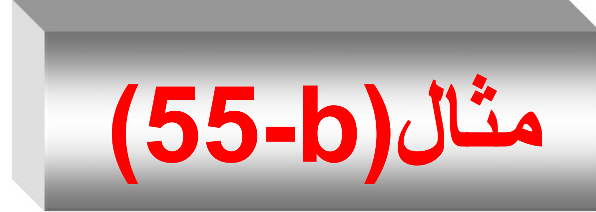
RESULT

```
The day is : Sunday
The day is : Monday
The day is : Tuesday
The day is : Wednesday
The day is : Thursday
The day is : Friday
The day is : Saturdays
Press any key to continue
```

طرائق (توابع) الصف Array

- يمتلك الصف Array عدداً من التوابع الستاتيكية (الساكنة) التي يمكن استخدامها في عمليات البحث Binarysearch والفرز Sort والوصول إلى العناصر وإنشاء نسخ فعالة من المصفوفة .
- يبين الجدول التالي بعض التوابع للصف Array .
- يبين البرنامج التالي استخدام الحلقة foreach واستخدام توابع الصف Array ، التابع Sort والتابع Reverse .

الطريقة	الوظيفة
BinarySearch( )	البحث عن عنصر في مصفوفة مرتبة أحادية البعد Pos = Array.BinarySearch(src , key)
Clear( )	تُهيئ أي عدد من عناصر المصفوفة بالقيمة صفر Array.Clear(src , 0 , src.Length)
Copy( )	تنسخ أي جزء من مصفوفة ما على مصفوفة أخرى Array.Copy(src , dest , dest.Length)
Sort( )	ترُتب عناصر المصفوفة أحادية البعد Array.Sort(src)
Reverse( )	تعكس ترتيب العناصر في مصفوفة أحادية البعد Array.Reverse(src)
Length	خاصية تُرجع طول المصفوفة Size = src.Length




```
using System;

class ArrayProg
 {
   public static void PrintMyArray(string[] weekDays )
    {
        foreach (string day in weekDays)
        {
           Console.WriteLine("The day is : {0} ", day);
          }
        } // end PintMyArray
```

```
public static void Main(string[] args)    {

 string[] weekDays = new string[7] { "Sunday", "Monday",
"Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" };
            Console.WriteLine(" the day is ");
            PrintMyArray( weekDays);
            Console.WriteLine(" Use Reverse ");
             Array.Reverse(weekDays);
             PrintMyArray(weekDays);
             Console.WriteLine(" Use Sort ");
             Array.Sort(weekDays);
             PrintMyArray(weekDays);

        } // end method main
    } // end class Name
```

RESULT

Use Reverse

The day is : Saturday
The day is : Friday
The day is : Thursday
The day is : Wednesday
The day is : Tuesday
The day is : Monday
The day is : Sunday

Press any key to continue

RESULT

the day is

The day is : Sunday
The day is : Monday
The day is : Tuesday
The day is : Wednesday
The day is : Thursday
The day is : Friday
The day is : Saturday. .

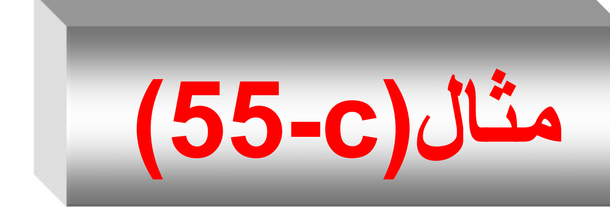
Press any key to continue

RESULT

Use Sort

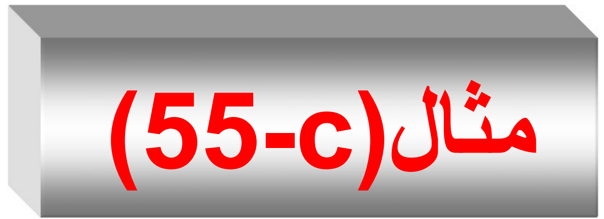
The day is : Friday
The day is : Monday
The day is : Saturday
The day is : Sunday
The day is : Thursday
The day is : Tuesday
The day is : Wednesday

Press any key to continue



- البرنامج التالي يستخدم الصف العشوائي Random لتوليد أسماء 12 شهراً بشكل عشوائي ضمن مصفوفة nameMonth والمطلوب :

1. أنشأ تابعاً اسمه PrintMyArray لطباعة عناصر المصفوفة nameMonth باستخدام الحلقة foreach .
2. أدخل عناصر المصفوفة nameMonth عن طريق توليد أسماء الأشهر باستخدام الصف Random .
3. استخدم الطريقة Sort( ) لترتيب عناصر المصفوفة .
4. استخدم الطريقة BinarySearch( ) من أجل البحث عن الشهر April .
5. استخدم الطريقة Reverse من أجل عكس عناصر المصفوفة .




```
sing System;

class ArrayProg
{
    public static void PrintMyArray(string[] nameMonth)
    {
        foreach (string month in nameMonth)
        {
         Console.WriteLine("The Month is : {0} ", month);
        }
    } // end PintMyArray
```

```
public static void Main(string[] args)
 {

  string[] nameMonth= new string[12];
  Random rnd = new Random();
  int nrt ;
  for (int i = 0; i < nameMonth.Length; i++)
  {

      nrt = rnd.Next(1, 13);
   DateTime dt = Convert.ToDateTime(nrt+"/"+ nrt+"/ 2017");

     nameMonth[i] = dt.ToString("MMMM"); ;
 } // end for
```

```
        Console.WriteLine(" the Month is ");
        PrintMyArray(nameMonth);
        Console.WriteLine(" Use Reverse ");
        Array.Reverse(nameMonth);
        PrintMyArray(nameMonth);
        Console.WriteLine(" Use Sort ");
        Array.Sort(nameMonth);
        PrintMyArray(nameMonth);

    } // end method main
} // end class Name
```

RESULT

Use Reverse

The Month is : February
The Month is : February
The Month is : June
The Month is : May
The Month is : March
The Month is : May
The Month is : August
The Month is : August
The Month is : September
The Month is : May
The Month is : September
The Month is : October
Press any key to continue

RESULT

Use Sort

The Month is : August
The Month is : August
The Month is : February
The Month is : February
The Month is : June
The Month is : March
The Month is : May
The Month is : May
The Month is : May
The Month is : October
The Month is : September
The Month is :
SeptemberPress any key to continue

RESULT

the day is

The Month is : October
th is : September
th is : May
th is : September
th is : August
th is : August
th is : May
th is : March
th is : May
th is : June
th is : February
th is : Februar
ny key to continue

الكلمة المحجوزة params

- تسمح الكلمة المحجوزة params بتمرير عدداً متغيراً من الوسائط ذات النوع الواحد إلى تابع ما .
- أما بالنسبة للتابع فيستقبل هذه الوسائط على شكل مصفوفة من ذات النوع .
- البرنامج التالي يحتوي على تابع اسمه Add يقبل عدداً متغيراً من الوسائط ذات النوع int ، ويقوم بحساب وطباعة مجموع عناصر المصفوفة باستخدام الكلمة المحجوزة params ، ويستخدم الحلقة foreach من اجل طباعة عناصر المصفوفة .

```
using System;
    class Program
    {
        static void Add(params int[] parray)
        {
            int sum = 0;
            foreach (int i in parray)
                sum = sum + i;
            Console.WriteLine("Sum =" + sum);
            Console.WriteLine(" parray Elements ");
            foreach (int i in parray)
            {
                Console.Write(" {0}  ", i);
            }
        }//end Add
```

```
static void Main(string[] args)
        {
            Add(11, 23, 73, 40, 5);

            Console.WriteLine();

        }//end main
    } // end class
```

RESULT

Sum =152
parray Elements
11   23   73   40   5
Press any key to continue . . .

2- المصفوفات متعددة الأبعاد
Multidimensional Arrays

- يمكن للمصفوفات في #C أن تكون متعددة الأبعاد ويمكن أن يكون كل بعد بحجم مختلف .
- يوجد نوعين من المصفوفات : المصفوفة المستطيلة ويرمز لها array[i,j] والمصفوفة غير المنتظمة ويرمز لها array [i][j] .
- **الاستعمال الشائع للمصفوفات متعددة الأبعاد هو تمثيل الجداول Tables, والجدول التالي يحتوي على معطيات مرتبة في صورة صفوف وأعمدة ولتمثيل الجدول نحتاج لبعدين الأول يمثل السطور (صف) والثاني يمثل الأعمدة.**
- ويبين الشكل التالي مصفوفة A مستطيلة تحتوى على ثلاثة صفوف وأربع أعمدة.

	Column 0	Column1	Column2	Column 3
Row 0	A[0,0]	A[0,1]	A[0,2]	A[0,3]
Row 1	A[1,0]	A[1,1]	A[1,2]	A[1,3]
Row 2	A[2,0]	A[2,1]	A[2,2]	A[2,3]

- يتم تمثيل أي عنصر في المصفوفة A على الصورة **A[i , j]**

حيث:

- **A** : اسم المصفوفة.
- **i** : رقم الصف الذي ينتمي إليه العنصر.
- **j** : رقم العمود الذي ينتمي إليه العنصر.

- نلاحظ أن كل العناصر الموجودة في الصف الأول مثلاً يكون الدليل الأول لها هو 0 وكل العناصر الموجودة في العمود الرابع يكون الدليل الثاني لها هو 3.
- يتم التصريح عن مصفوفة a تحتوى على x صف و y عمود للمصفوفة المستطيلة كما يلي :

```
int [ , ] a = new int [x,y];
```

- حيث x عدد الصفوف و y عدد الأعمدة .
- يمكن إسناد قيم أولية للمصفوفة المستطيلة المتعددة الأبعاد عند التصريح عنها وذلك كما يلي :

```
int [ , ] b = {  {1,2} , {3,4}  };
```

- **حيث:**

b[1,1]=4 , b[1,0]=3 , b[0,1]=2 , b[0,0]=1

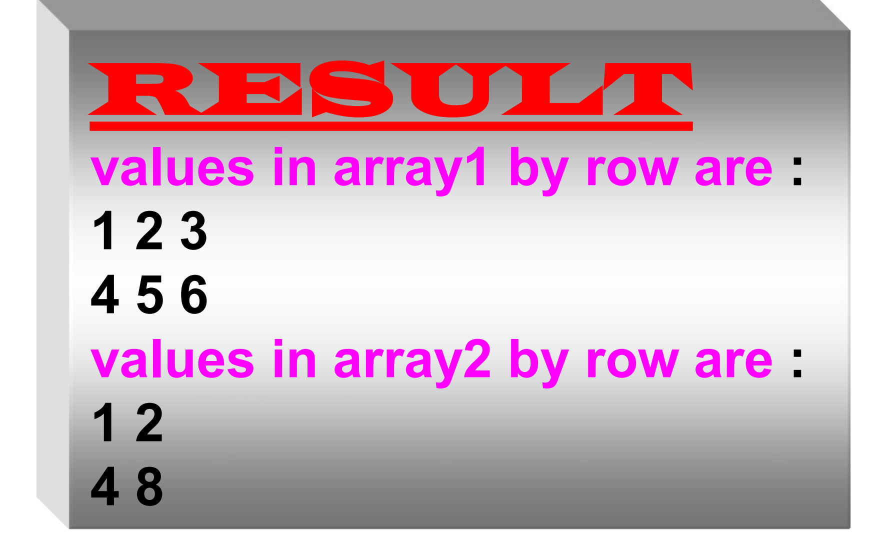
- أيضاً نلاحظ في المصفوفة المستطيلة متعددة الأبعاد إذا تم إسناد قيم أولية لا يتوافق عددها مع حجم المصفوفة فإن المترجم سيملأ بقية العناصر أصفار.
- ويوضح البرنامج التالي كيفية إسناد قيم أولية للمصفوفات متعددة الأبعاد عند التصريح عنها:




```
// initializing multidimensional arrays
using System ;
 namespace   Example56 {
class DimArry0 {
public static void Main(string [ ] args ) {
int [ , ]array1 = {  {1, 2, 3}, {4, 5, 6} };
int[ , ] array2 = { { 1, 2 }, { 4 , 8  }  };
Console.WriteLine("values in array1 by row are : ");
for (int i=0; i< 2 ; i++) {
  for (int j=0; j< 3; j++)
 Console.Write(array1[i,j] +" ");
 Console.WriteLine();  } //end for
```

```
//Continued
Console.WriteLine("values in array2 by row are : ");
  for (int i=0; i< 2 ; i++) {
  for (int j=0; j< 2; j++)
 Console.Write(array2[i,j] +" ");
 Console.WriteLine();  } //end for
}//End main
} // end class Name
 } //end namespace
```





المصفوفات غير المنتظمة

- كل صف في المصفوفة متعددة الأبعاد هو مصفوفة بحد ذاته , وهكذا فإنه من الممكن للصفوف أن تملك أطوال مختلفة .
- تعرف المصفوفة من هذا النوع بالمصفوفة غير المنتظمة , فيما يلي مثال عن إنشاء مصفوفة غير منتظمة :

```
int [ ][ ] array = new int [5][ ] ;
```

- لنفرض لدينا القيم الابتدائية للمصفوفة غير المنتظمة :

{ { 1,2,3,4,5} , { 2,3,4,5} , { 3,4,5} , { 4,5} , { 5} }

- لذلك فإن طول array[0].Length يساوي 5 , وطول array[1].Length يساوي 4 , وطول array[2].Length يساوي 3 , وطول array[3].Length يساوي 2 , وطول array[4].Length يساوي 1 .

- إذا كانت القيم التي ستخزن في المصفوفة غير المنتظمة معروفة فإنه يمكن صنع مصفوفة غير منتظمة باستخدام الصيغة التالية :

```
int [ ][ ] array = new int [5][ ] ;
array [0] = new int [5] { 1,2,3,4,5} ;
array [1] = new int [4] { 2,3,4,5} ;
array [2] = new int [3] { 3,4,5} ;
array [3] = new int [2] { 4,5} ;
array [4] = new int [1] { 5} ;
```

يمكننا الآن إسناد قيم عشوائية إلى هذه المصفوفة باستخدام الحلقة التالية:

```
Random rnd = new Random();
  for (int i = 0; i < array1.Length; i++)
      for (int j = 0; j < array1[i].Length; j++)

          array1[i][j] = rnd.Next(1, 13);


  for (int i = 0; i < array1.Length; i++)
      for (int j = 0; j < array1[i].Length; j++)
  {
    Console.WriteLine("The Element is : {0} ",array1[i][j]);
  }
```

- تحتاج الصيغة new int [5][ ] إلى تحديد الدليل الأول من أجل إنشاء مصفوفة ، لأن الصيغة التالية خاطئة

new int [ ][ ]

مثال(57)

- البرنامج التالي يوضح كيفية اسناد قيم ابتدائية لمصفوفة منتظمة array2 وطباعة عناصرها واستخدام Length للوصول لحجم المصفوفة .
- اسناد قيم ابتدائية لمصفوفة غير منتظمة array1 وطباعة عناصرها واستخدام Length للوصول لحجم المصفوفة .
- استخدام الصف Random لتوليد قيم عشوائية واسنادها للمصفوفة المنتظمة array1 وطباعة عناصرها واستخدام Length للوصول لحجم المصفوفة .
- الخرج يجب أن يكون على شكل مصفوفة ثنائية البعد .




```
using System ;

class DimArry0 {
public static void Main(string [ ] args ) {
int [ ][]array1 =new int[2][];
    array1[0]=new int[3] {1, 2, 3};
    array1[1]=new int[2]  {4, 5} ;
int [ , ]array2 = { { 1, 2 }, { 4 , 8  }  };

Console.WriteLine("values in array1 by row are : ");
for (int i = 0; i < array1.Length; i++) {
    for (int j = 0; j < array1[i].Length; j++)
        Console.Write(array1[i][j] + " ");
        Console.WriteLine();
 } //end for
```

```
//Continued
Console.WriteLine("values in array2 by row are : ");
  for (int i=0; i< 2 ; i++) {
  for (int j=0; j< 2; j++)
 Console.Write(array2[i,j] +" ");
 Console.WriteLine();
} //end for
//----------------------
  Console.WriteLine("values in NEW array1 by row are : ");
  Random rnd = new Random();
  for (int i = 0; i < array1.Length; i++)
      for (int j = 0; j < array1[i].Length; j++)
          array1[i][j] = rnd.Next(1, 13);
```

```
// foreach (int s in array1)
   for (int i = 0; i < array1.Length; i++)
   {
       for (int j = 0; j < array1[i].Length; j++)
           Console.Write("{0,2} ", array1[i][j]);
         Console.WriteLine();
   }
//-------------------
}//End main
} // end class Name
```

RESULT

Values in array1 by row are :
1 2 3
 4 5
Values in array2 by row are :
1 2
4 8
Values in NEW array1 by row are :
 1 7 4
 1 5.
Press any key to continue

إدخال وإخراج المصفوفة ثنائية البعد

- يمكن إدخال عناصر المصفوفة من قبل المستخدم
- **وأيضاً يمكن إخراج (طباعة) عناصر المصفوفة ثنائية البعد بواسطة حلقتين (حلقتي for , أو while , أو do/while , أو خليط بينهم) .**
- قبل استخدام المصفوفة يمكن تصفير المصفوفة وذلك بإعطاء قيم ابتدائية تساوي الصفر .
- يبين المثال التالي كيفية إدخال وإخراج عناصر المصفوفة a[10,5].



- **اكتب برنامجاً بلغة #C يقوم بما يلي :**
  - إدخال درجات 10طلاب في خمس مقررات وتخزينها في مصفوفة ثنائية البعد [a[10,5 .
  - حساب وطباعة معدل كل طالب في المقررات الخمسة .
  - حساب وطباعة المعدل الأكبر .



```
// multidimensional arrays
using System ;
namespace   Example58 {
class DimArry1 {
public static void Main(string [ ] args ) {
int [ , ] Mark;
Mark = new int[10,5] ;
  int ormax ;
  double  sum=0.0 , maxav ;
  double  [ ]avr;
  avr =new double [10];
```

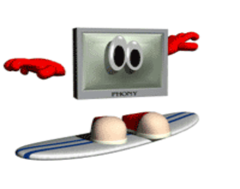
```
Console.WriteLine("Enter Mrka[ , ] values  :  ");
   for (int  i=0; i< 10 ; i++ )
   for (int  j=0; j< 5 ; j++ ){
int x = int.Parse( Console.ReadLine() );
   Mark[i,j]=x;
   }
//Continued
for (int  k=0; k<10; k++) {
   sum=0.0 ;
 for (int  j=0; j<5; j++) {
    sum=sum + Mark[ k , j ] ; }
        avr[ k ]=sum / 5 ;
}
```

```
maxav = avr[0];
ormax =1;
for (int i2=0; i2<10; i2++)   {
    if( avr[ i2 ] > maxav ) {
        maxav = avr[ i2 ] ;
         ormax = i2+1 ;          }
}
for (int i1=0; i1<10; i1++) {
Console.WriteLine(" avr ["+i1+ "]=" +avr[i1] );   }
Console.WriteLine(" The greatest average is " + maxav );
Console.WriteLine(" The greatest number is "+ ormax )
}//End main
} // end class Name
 } //end namespace
```

RESULT

```
av[0]=187.5        av[1]=187
av[2]=180          av[3]=189.5
av[4]=181.5        av[5]=165
av[6]=195          av[7]=197.5
av[8]=176          av[9]=187
The greatest average is  197.5
The greatest number is    7

Press any key to continue
```



تم استخدام التابع printArray( ) لطباعة عناصر المصفوفة




```
// initializing multidimensional arrays
using System ;
class DimArry {
public static void Main(string [ ] args ) {
  int[][] array1 = new int[2][];// {{1, 2, 3}, {4, 5, 6}};
    array1[0] = new int[3] { 1, 2, 3 };

array1[1] = new int[3] { 4, 5, 6 };

int[][] array2 = new int[2][]; // { { 1, 2 } , { 4 } };
array2[0] = new int[2] { 1, 2 };

array2[1] = new int[1] { 4 };
```

```
onsole.WriteLine("values in array1 by row are : ");
     printArray( array1 ) ;
//Continued
Console.WriteLine("values in array2 by row are : ");
  printArray(array2);
}//End main
public static void printArray(int [ ][ ] array) {
  for (int i=0; i< array.Length ; i++) {
  for (int j=0; j< array[ i ].Length ; j++)
 Console.Write(array[i][j] +" ");
 Console.WriteLine();
  } //end for

   }//End function
} // end class Name
```

RESULT

values in array1 by row are :
1 2 3
4 5 6
values in array2 by row are :
1 2
4
Press any key to continue

مثال(60)

- اكتب برنامجاً بلغة ++C يقوم بما يلي :
  - إدخال 12عدداً صحيحاً , ثم يتم تخزينها في مصفوفة ثنائية البعد [3,4]a .
  - طباعة عناصر المصفوفة a .
  - حساب المتوسط الحسابي لعناصر المصفوفة وطباعته .
- استخدم التوابع في البرنامج كما يلي :
  - التابع ( )printArray لطباعة عناصر المصفوفة .
  - التابع ( )printArray1 لحساب مجموع عناصر المصفوفة وحساب المتوسط الحسابي

مثال(60)

```
//  multidimensional arrays
 using System ;
 namespace   Example59 {
class DimArry2 {
public static void Main(string [ ] args ) {
int [ , ] Arry;
Arry = new int[3,4] ;
 Console.WriteLine("Enter Arry [ , ] values  :  ");
   for (int  i=0; i<3; i++ )
   for (int  j=0; j<4; j++ ){
    int x = int.Parse( Console.ReadLine() );
   Arry[i,j]=x;
   }
   printArray(Arry);
   printArray1(Arry);
   }//End main
```

```
//Continued
public static void printArray ( int [ , ] a) {
  for (int i=0; i<3; i++){
  for (int j=0; j<4; j++)
Console.Write ( a[ i , j ] +"  " );
 Console.WriteLine();}
}//End printArray
public static void printArray1(int [ , ] a) {
   int sum=0;
    for (int i=0; i<3; i++)
    for (int j=0; j<4; j++)
     sum = sum + a[ i , j ] ;
    double  avr = sum/12;
 Console.WriteLine(" avr = "+  avr );
   }//End printArray1
} // end class Name
 } //end namespace
```

RESULT

Enter a[ ][ ] values :

11 22 33 44 55 66 77 88 99 12 13 14

11 22 33 44
55 66 77 88
99 12 13 14

Avr = 44
Press any key to continue

مثال(61)

- اكتب برنامجاً بلغة #C يقوم بما يلي :
  - إدخال عناصر المصفوفة [a[3,3 وهي مصفوفة ثنائية البعد ذات أعداد حقيقية .
  - طباعة عناصر المصفوفة a .
  - حساب المتوسط الحسابي للأعداد التي تقع فوق القطر الرئيسي للمصفوفة وطباعته .
- استخدم التوابع في كتابة البرنامج

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

```
//  multidimensional arrays
using System ;
 namespace   Example60 {
class DimArry3 {
public static void Main(string [ ] args ) {
int [ , ] A;
A = new int[3,3] ;
 Console.WriteLine("Enter A[ , ] values  :  ");
  for (int  i=0; i<3; i++ )
  for (int  j=0; j<3; j++ ){
   int x = int.Parse( Console.ReadLine() );
  A[i,j]=x;
  }
  printArray1(A);
  printArray(A);
  }//End main
```

```
public static void printArray ( int [ , ] a )
{
 int sum=0 ;
   int n = 0 ;
   for (int i=0; i<3; i++) {
   for (int j=0; j<3; j++)
   if( i<= j ) {
   sum=sum + a[ i , j ] ;
    n = n+1 ; }
  }
  double   avr = sum / n ;
 Console.WriteLine(“\n avr = "+  avr ) ;

  }//End printArray1
```

```
public static void printArray1(int [ , ] a) {
    Console.WriteLine( " Print Array");
   for (int i=0; i<3; i++){
   for (int j=0; j<3; j++)
Console.Write (  a[ i , j ] +" ");
Console.WriteLine();
   }
}//End printArry


   } // end class Name
 } //end namespace
```

RESULT

Enter a[,] values :
11 22 33 55 66 77 99 12 13

11 22 33
55 66 77
99 12 13
Avr = 44

مثال(62)

- **اكتب برنامجاً بلغة C# يقوم بتصحيح مؤتمت لامتحان مقرر البرمجة لخمس طلاب بحيث يكون عدد الأسئلة المؤتمتة عشرة ويكون عدد الأجوبة المحتملة لكل سؤال أربعة (A,B,C,D) والمطلوب :**
  - إدخال عدد الطلاب وعدد الأسئلة، بحيث يتم تخزينها في مصفوفة محرفية ثنائية البعد answers [5,10] ،أو إسناد قيم ابتدائية للمصفوفة تمثل أجوبة الطلاب .
  - إدخال (أو إسناد) الـ Key ، وهو مصفوفة محرفية أحادية البعد تمثل الأجوبة الصحيحة key[10].
  - أنشئ تابعاً اسمه Results بوسيطين الوسيط الأول مصفوفة ثنائية البعد والوسيط الثاني مصفوفة أحادية البعد ، يقوم بحساب مجموع عدد الأسئلة الصحيحة لكل طالب .
  - أنشئ تابعاً اسمه MaxResults يقوم بحساب أعلى درجة حصل عليها أحد الطلاب .
- التابع main يستدعي التابعين ويطبع درجات الطلاب وأعلى درجة .

```
using System;
  class Program {
    static void Main(string[] args)
    {
      /* char [ , ]answers = { { 'A' , 'A' , 'D' , 'A' , 'C' , 'A' , 'A' , 'B' , 'B' , 'A' },
                              { 'A' , 'D' , 'C' , 'A' , 'C' , 'D' , 'D' , 'B' , 'C' , 'A' },
                              { 'A' , 'B' , 'D' , 'C' , 'C' , 'A' , 'D' , 'C' , 'B' , 'C' },
                              { 'A' , 'B' , 'D' , 'A' , 'C' , 'A' , 'D' , 'B' , 'B' , 'A' },
                              { 'A' , 'C' , 'A' , 'C' , 'A' , 'D' , 'D' , 'C' , 'A' , 'A' },
            }; */
    char ch;
         char [ , ]answers=new char[5,10] ;
         Console.WriteLine("Enter answers ");
   for ( int i=0 ; i< 10 ; i++ )
   for ( int j=0 ; j< 5 ; j++ ){
         ch = char.Parse( Console.ReadLine() );
      answers[i , j]=ch ;
   } //end loop
```

```
  char [ ] keys = {'A','B','D','C','C','A','D','B','B','A' };
     Results (  answers , keys ) ;
  }//end main
public static void  Results ( char[ , ]answers,char [ ]keys ) {
 int [ ] correctCount1= new int[5] ;
for ( int i=0 ; i< 5 ; i++ ){
  int correctCount =0 ;
    for (int j = 0; j < 10; j++)
    {
      if(answers [i,j] == keys[j])
       correctCount++ ;
    }//end for j
  correctCount = correctCount *10 ;
  correctCount1[ i]=correctCount ;
Console.WriteLine("student"+(i+1)+" 's correct Count is "+correctCount );
} //end for i
Console.Write("Max correct Count Of student="+MaxResults(correctCount1) );
```

```
public static int MaxResults(int[] max)
  {
   int xmax=max[0] ;
   for ( int i=0 ; i< max.Length ; i++ ){
   if( max[i] > xmax )
 xmax = max[i] ;
    }// end for
   return xmax ;
  }// end MaxResults
 }// end class
```

RESULT

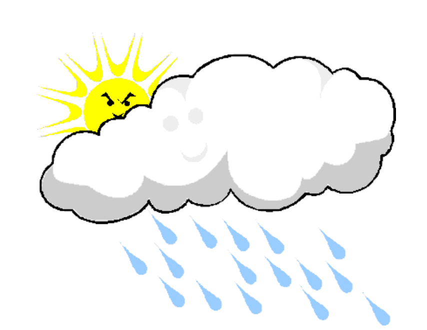
student 1 's correct Count is 70
student 2 's correct Count is 50
student 3 's correct Count is 80
student 4 's correct Count is 90
student 5 's correct Count is 40
Max correct Count Of student = 90
Press any key to continue

تمارين عامة(3)

1. اكتب برنامجاً بلغة #C يقوم بما يلي :
   - يدخل 20 عدداً حقيقياً ويخزنها في مصفوفة أحادية البعد A[20].
   - ويحتوي البرنامج على تابعين:
     - التابع الأول اسمه (   )SumAve مهمته حساب المتوسط الحسابي لعناصر المصفوفة.
     - أما التابع الثاني اسمه EvenArry مهمته حساب مجموع الأعداد الزوجية في المصفوفة .
   - التابع main يستدعي التابعين ويطبع المتوسط ومجموع الأعداد الزوجية .

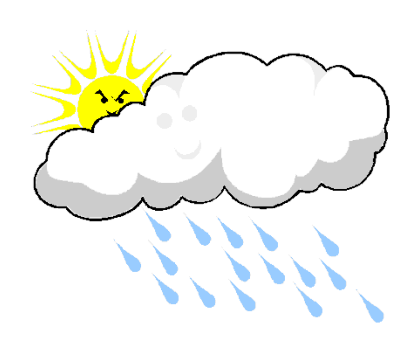
2. اكتب برنامجاً بلغة C# يقوم بما يلي :

– يدخل 20 عدداً حقيقياً ويخزنها في مصفوفة أحادية البعد A[20].
– ويحتوي البرنامج على تابعين:
  • التابع الأول اسمه MaxArry( ) مهمته حساب العنصر الأكبر للمصفوفة .
  • أما التابع الثاني اسمه OddArry مهمته حساب مجموع الأعداد الفردية في المصفوفة .
– التابع main يستدعي التابعين ويطبع العنصر الأكبر ومجموع الأعداد الفردية .



3. أكتب برنامجاً بلغة C# يقوم بما يلي:

- إدخال 10 أعداد صحيحة وتخزينها في مصفوفة أحادية البعد a[10] .
- يستخدم تابع من النوع void اسمه sort لترتيب الأعداد المدخلة بشكل تصاعدي.
- التابع main يستدعي التابع sort ويطبع طباعة الأعداد المدخلة بعد عملية الترتيب .



4. أكتب برنامجاً بلغة #C يقوم بما يلي:

- إدخال 9 أعداد صحيحة وتخزينها في مصفوفة ثنائية البعد x[3,3] .
- طباعة عناصر المصفوفة x[3,3] على شكل مصفوفة مربعة
- يستخدم تابع من النوع void لإيجاد كافة الأعداد الأولية من بين عناصر المصفوفة المُدخلة ، ومن ثم طباعة هذه الأعداد وأيضاً طباعة عددها.
- التابع main يستدعي التابع لطباعة النتائج .



the end

The End

The End

The End

The End

The End

The End

The End